EAGLES

Expert Advisory Group on
Language Engineering Standards

EAGLES

Evaluation of Natural Language Processing Systems

EAGLES DOCUMENT EAG–EWG–PR.2

Version of September, 1995

Contents

Chapter 1

Introduction

1.1 Contributors

Note: This report is a collective effort. It builds upon the contributions from the core Evaluation Working Group and from its sub-groups. The core group as well as the sub-groups have slightly changed membership over the three year period of this EAGLES initiative thus, as the report is the result of contributions from both the early and the late members, we have given the extensive list of those who contributed over time below. Finally, we should mention that this report builds heavily on the preliminary report of July 1994.

1.1.1 Authors

KING, Maghi	(Chairman)
ISSCO	
54, route des Acacias	Tel: +41.22.705.7114
CH-1227 Geneva	Fax: +41.22.300.1086
Switzerland	E-mail: Margaret.King@issco.unige.ch
MAEGAARD, Bente	(Host; Chairman, Writers' Aids)
Centre for Language Technology	
Njalsgade 80	Tel: +45.35.32.90.74
DK 2300 Copenhagen	Fax: +45.35.32.90.89
Denmark	E-mail: bente@cst.ku.dk
SCHÜTZ, Jörg	(Acting Chairman, Knowledge Management Systems)
IAI	
Martin Luther Str. 14	Tel: +49.681.38951
D-66111 Saarbrücken	Fax: +49.681.397482
Germany	E-mail: joerg@iai.uni-sb.de

TOMBE, Louis des	(Acting Chairman, Translators' Aids)
Research Institute for	
Language and Speech	
Trans 10	Tel: +31.30.536.049
NL-3512 JK Utrecht	Fax: +31.30.536.000
The Netherlands	E-mail: destombe@let.ruu.nl
BECH, Annelise	(Editor, preliminary report)
Centre for Language Technology	
Njalsgade 80	Tel: +45.35.32.90.76
DK-2300 Copenhagen	Fax: +45.35.32.90.89
Denmark	E-mail: cstadm@cst.ku.dk
NEVILLE, Anne	(Editor, final report)
Centre for Language Technology	
Njalsgade 80	Tel: +45.35.32.90.76
DK-2300 Copenhagen	Fax: +45.35.32.90.89
Denmark	E-mail: anne@cst.ku.dk
ARPPE, Antti	(Translators' Aids)
Institute of Industrial Management	
Helsinki University of Technology (HUT)	
Office U503	
Otakaari 1 M	Tel: +358.0451.3088
FIN-02150 Espoo	Fax: +358.0451.3095
Finland	E-mail: Antti.Arppe@hut.fi
BALKAN, Lorna	(Writers' Aids)
Department of Language and Linguistics	
University of Essex	
Wivenhoe Park	Tel: +44.206.872091
Colchester CO4 3SQ	Fax: +44.206.872085
UK	E-mail: balka@sx.ac.uk
BRACE, Colin	(Translators' Aids)
Language Industry Monitor	
Eerste Helmersstraat 183	Tel: +31.20.685.0462
NL-1054 DT Amsterdam	Fax: +31.20.685.4300
The Netherlands	E-mail: colinb@paramount.nikhefk.nikhef.nl
BUNT, Harry	(Translators' Aids)
Institute for Language Technology	
and Artificial Intelligence	
University of Tilburg	
PO Box 90153	Tel: +31.13.66.3060
NL-5000 LE Tilburg	Fax: +31.13.66.2537
The Netherlands	E-mail: bunt@kub.nl

CARLSON, Lauri	(Translators' Aids)
Department of Translation Studies	
University of Helsinki	
PL 94	Tel: +358.51.8252.206
45131 Kouvola	Fax: +358.51.8252.251
Finland	E-mail: lcarlson@ling.helsinki.fi
DOUGLAS, Shona	(Writers' Aids)
Human Communication Research Centre	
University of Edinburgh	
2 Buccleuch Place	Tel: +44.131.650.4665
Edinburgh EH8 9LW	Fax: +44.131.650.4587
UK	E-mail: shona@cogsci.ed.ac.uk
HÖGE, Monika	(Translators' Aids)
Department of Translation Studies	
University of Helsinki	
Home office:	
Gustav-Adolf Str. 6	Tel: +49.8171.31132
D-82538 Geretsried	
Germany	E-mail: 100340.2435@compuserve.com/mhoge@babel.helsinki.fi
KRAUWER, Steven	(Translators' Aids)
Research Institute for	
Language and Speech	
Trans 10	Tel:
NL-3512 JK Utrecht	Fax: +31.30.536.000
The Netherlands	E-mail: krauwer@let.ruu.nl
MANZI, Sandra	(Writers' Aids)
ISSCO	
54, route des Acacias	Tel: +41.22.705.7116
CH-1227 Geneva	Fax: +41.22.300.1086
Switzerland	E-mail: Sandra.Manzi@issco.unige.ch
MAZZI, Cristina	(Writers' Aids)
Centre for Language Technology	
Njalsgade 80	Tel: +45.35.32.90.65
DK 2300 Copenhagen	Fax: +45.35.32.90.89
Denmark	E-mail: cstadm@cst.ku.dk
SIELEMANN, Ann June	(Writers' Aids)
Centre for Language Technology	
Njalsgade 80	Tel: +45.35.32.90.76
DK-2300 Copenhagen	Fax: +45.35.32.90.89
Denmark	E-mail: cstadm@cst.ku.dk

STEENBAKKERS, Ragna	(Translators' Aids)
Department of Romance Languages and Cultures	
Faculty of Arts	
University of Utrecht	
Kromme Nieuwegracht 29	Tel: +31.30.536236
3512 HD Utrecht	Fax: +31.30.536167
The Netherlands	E-mail: Ragna.Steenbakkers@let.ruu.nl

1.1.2 Other contributors

FALKEDAL, Kirsten	(Translators' Aids)
ISSCO	
54, route des Acacias	Tel: +41.22.705.7116
CH-1227 Geneva	Fax: +41.22.300.1086
Switzerland	E-mail: kirsten@divsun.unige.ch
HAVENITH, Roger	(CEC Representative)
DG XIII–E/4	
Commission of the European Communities	
Bâtiment Jean Monnet	Tel: +352.4301.32886
Plateau du Kirchberg	Fax: +352.4301.34999
L-2920 Luxembourg	E-mail: roger_havenith@eurokom.ie
McNAUGHT, John	(Co-Chief Editor, EAGLES)
Centre for Computational Linguistics	
UMIST	
PO Box 88	
Sackville Street	Tel: +44.61.200.3098
Manchester M60 1QD	Fax: +44.61.200.3099
UK	E-mail: jock@ccl.umist.ac.uk
NORMIER, Bernard	(Adviser)
GSI ERLI	
1, place des Marseillais	Tel: +33.1.4893.8121
94227 Charenton-le-Pont Cedex	Fax: +33.1.4375.7979
France	E-mail: bernard.normier@erli.gsi.fr
PASTEUR, Olivier	(Translators' Aids)
Computer-assisted Translation	
and Terminology Service	
World Health Organization	
Av. Appia 20	Tel: +41.22.791.23.17
CH-1211 Geneva 27	Fax: +41.22.791.07.46
Switzerland	E-mail: pasteur@who.arcom.ch
PEROT Jean Jacques	(Adviser)
SITE-EUROLANG	
BP 35	Tel: +33.1.4513.0500
94701 Maisons Alfort Cedex	Fax: +33.1.4513.0559
France	E-mail: jjacques.perot@site-maisons-alfort.fr
THOMPSON, Henry S.	
Human Communication	
Research Centre	
University of Edinburgh	
2 Buccleuch Place	Tel: +44.31.650.4665
Edinburgh EH8 9LW	Fax: +44.31.650.4587
UK	E-mail: ht@cogsci.ed.ac.uk

WEHRLI, Tamara	(Secretary, document production)
ISSCO	
54, route des Acacias	Tel: +41.22.705.7115
CH-1227 Geneva	Fax: +41.22.300.1086
Switzerland	E-mail: issco-admin@divsun.unige.ch

WINSKI, Richard	(Liaison SLWG and EWG)
VOCALIS Ltd	
Chaston House, Mill Court	
Station Road	
Great Shelford	Tel: +44.223.846177
Cambridge	Fax: +44.223.846178
UK	E-mail: richard@vocalis.demon.co.uk

1.2 Guide to the report

1.2.1 About this report

This report represents the work done by the EAGLES Evaluation Working Group in the first 30 months of its existence. Although it is a final report on the work done in that period, there is no sense in which it is a report on a fully completed and polished piece of work, presenting a thoroughly worked out and integrated methodology for the design of evaluations. The problem area we have been attacking is vast, and there was little previous work available as a starting point. Nevertheless, the long term objective of the group is to identify and specify the components of a compendium of evaluation criteria and associated techniques, together with guidelines for their use, from which the individual evaluation user can select those techniques which are relevant to his purpose.

The work of the group has been entirely concerned with natural language processing systems and products. Within this broad area, the group initially selected three types of system for study, writers' aids, translators' aids and knowledge management systems. Work has been narrowly focused on a particular type of evaluation, namely *adequacy evaluation*, which is the activity of assessing the adequacy of a system with respect to some intended use of that system. The resources available to the group were rather unevenly distributed. In consequence, writers' aids and translators' aids have received rather different treatment from the knowledge management systems, where all that could be done was to set up a set of desirables for such systems or, in other words, to work out a requirements specification from which criteria for evaluation could be deduced.

In what follows, although some sections are, we hope, pre-final in the sense that no complete re-thinking of the basics should be necessary, nothing is to be thought of as totally definitive. With this in mind, we welcome all comments and suggestions, which can be sent either to ewgeditor@tnos.ilc.pi.cnr.it (EWG Editor), ceditor@tnos.ilc.pi.cnr.it (Editorial Board) or eagles@ilc.pi.cnr.it (EAGLES secretariat). The reader is equally welcome to contact individual members concerning particular points of interest.

1.2.2 How to read it

The report contains a main body, which sets out the general framework proposed by the group and a certain amount of background material, accompanied by substantial appendices which report on the work in particular areas.

The main body starts by recapitulating the starting point, an existing Standard, ISO 9126, and explaining the extensions to that Standard made by the group. Most of these extensions come from an attempt to think of evaluation not in terms of the desires of one particular specific user/customer, whose needs and constraints can be built into the evaluation, but in terms of a general evaluation methodology which, catering potentially for the desires of a wide variety of different classes of user, can be tailored in any given case to reflect specific desires.

The section 2 describes a first attempt to make such an evaluation concrete in the form of a parametrized test-bed. The basic idea here is that it should be possible to construct: descriptions of products (the objects of evaluation); descriptions of classes of users (the customers of evaluation); and descriptions of attributes of systems potentially of interest to classes of users, coupled with metrics which, when applied to a product, would provide a value for that product for each attribute. Then, when an evaluation of a particular product or class of products is to be performed on behalf of a particular class of users, the desires of that class of users can be used to pick out the attributes and metrics pertinent to the specific case. Although some of the metrics lead to automated tests, others are by their nature not completely automatable. In these cases, the test-bed produces a set of instructions for the human on how to conduct and report on the test.

In order to be able to produce such a parametrized test-bed, the descriptions of products, of users and of tests need to be described in formal terms. This section therefore also contains some preliminary work on a formalisation in terms of attribute-value structures of the sort familiar from work in computational linguistics.

The characteristics of classes of users need to be discovered and formalized. The following section (2.2) describes and discusses some preliminary work on techniques for doing this, drawing inspiration from recent work on requirements analysis in the software engineering field.

The final parts of the framework model chapter (2.5 and 2.6) review experience in software engineering and the methods used there for carrying out an evaluation. The emphasis is on experiment design, and the advantages and disadvantages of a variety of measures and methods are discussed.

The second part of the main body of the report contains background material. The first section (3.1) reports on the use of consumer reports in other areas. As the reader will discover, the consumer report paradigm has served as an important part of the framework model. Next, brief summaries are given of work within some of the LRE (Language Research and Engineering) projects which have collaborated with or made use of EAGLES work (3.2) An account of relevant previous work on evaluation (3.3) is also given.

It is intended that the main body of the report can be read independently of the appendices, although it should be remembered that the attempts to put the framework model of the main body into practical application are to be found in the appendices, and that the reader might therefore find it useful from time to time to look for detailed examples there.

The first appendix (A) gives an overview of relevant ISO terms and guidelines.

The second appendix (B) provides a selection of methods for the measurement of software, focusing on the special problems that software evaluation poses in the NLP area.

Then follows an appendix (C) describing investigations into issues and methods for user profiling and requirements analysis for language engineering evaluation. The appendix is a fuller version of the section on requirements in the main body of the report.

The appendices then cover first the application of the framework to the evaluation of writers' aids(D) and translators' aids (E) These two appendices are substantial attempts to work out detailed applications of the framework. They are however far from being complete: although much work has been done on evaluating the functionalities of the products considered, much less has been done on characteristics such as usability where there were less solid starting points.

The appendix on writers' aids is supplemented by an appendix giving a detailed account of testing carried out on grammar checkers (I). The appendix on translators' aids will be supplemented in the near future by detailed accounts of testing carried out on translation memories.

The next appendix (J) reports on the rather more limited work we have been able to do on the evaluation of knowledge management systems. As we have already mentioned, this work takes the form of working out a set of requirements specifications for such systems, from which evaluation criteria may subsequently be deduced.

The final appendix (K) is also rather different in nature. It consists of a detailed descriptive study of a very large translation service, that of the European Commission. It was decided to carry out such a study after earlier work on user profiling, reported on in the interim report, (EAGLES, 1994), led to the realisation that a translation service as large and as complex as that of the Commission did not fit easily into any neat pigeonhole designed to hold the profile of a typical user.

We close this guide by reminding the reader once again that they are reading a report on work in progress: it is never fully fashioned, frequently incomplete and sometimes even fragmentary. We would be grateful for reactions and indeed for help, as work on standardisation is only validated and advanced through feedback and collaboration. Comments may be communicated in various forms:

- via WWW with the URL: http://www.ilc.pi.cnt.it/EAGLES/modulo.html

- via the contact e-mail addresses
  - ewgeditor@tnos.ilc.pi.cnr.it (EWG Editor)
  - ceditor@tnos.ilc.pi.cnr.it (Editorial Board)
  - eagles@ilc.pi.cnr.it (EAGLES secretariat)
- by post/fax to Tarina Ayazi at the EAGLES secretariat address or to the EWG Editor at her address, using the form at the back of the printed document (EAGLES Comments Cover Form)

1.3 Types of systems considered

1.3.1 Long term

As already mentioned, the long term objective of the group is to identify and specify the components of a compendium of evaluation criteria and associated techniques, together with guidelines for their use, from which the individual evaluation user can select those techniques which are relevant to his purpose.

Once this compendium exists it can be used as a starting point to determine a standard against which new and existing products can be measured and compared. It may also be useful for systematising the definition of a goal or set of goals for new projects and for measuring progress towards those goals.

Although the group will seek to keep abreast of the development of evaluation and assessment methods for speech technology, and may indeed find some inspiration there, the primary emphasis is on evaluation of systems applied to written text. Within that constraint, the group will in the long term consider a wide range of application areas for natural language processing systems, ranging from spelling, grammar and style checkers at one end of the market to natural language interfaces and machine translation systems at the other.

Application areas to be considered will include:

- Writers' aids
  - spelling checkers
  - grammar checkers
  - style checkers
  - on-line monolingual and bilingual dictionaries, and dictionary access systems
- Information management tools
  - automatic indexing systems
  - text retrieval systems
  - information retrieval systems with a natural language component
  - authoring aids with an information management component
- Natural language front ends for
  - information systems
  - data base systems
  - computer systems
- Translators' aids

  - translation memories
  - specialised workstations
  - terminology management data bases
  - electronic monolingual and multilingual dictionaries, and dictionary access systems
  - multilingual thesauri

- Machine translation systems
- Natural language generation systems
- Message understanding systems

This list is not intended as either definitive or exclusive. Note too that although the list is organised around the notion of an application area, this is not meant to imply that the group will have no interest in the evaluation of basic or applied research.

1.3.2 Short term

In the more immediate short term, limited resources and a desire to produce concrete results have led us to focus on adequacy evaluation of market or near-market products in two areas, writers' aids and translators' aids. Each of the two groups created to work on these areas has concentrated on one particular aspect of the overall task of setting up an evaluation methodology according to the guidelines outlined in chapter 2.

Writers' aids: (appendix D) This group has concentrated on the methodology to be followed in elaborating measures and methods. A relatively small number of attributes has been taken into account, but the measures and methods for those attributes have been worked out in concrete detail.

Translators' aids: (appendix E) This group has concentrated on the methodology for establishing pertinent attributes, i.e. for the construction of consumer report grids.

Although the focus of each group is directed to one particular point of the general framework, care has been taken to ensure that work across the two groups is consistent with the framework as a whole.

Chapter 2

The Framework Model

2.1 The ISO 9126 Standard

2.1.1 ISO 9126

The EAGLES work takes as its starting point an existing Standard, ISO 9126, which is concerned primarily with the definition of quality characteristics to be used in the evaluation of software products. ISO 9126 sets out six quality characteristics, which are intended to be exhaustive. From this it follows that each quality characteristic is very broad. We shall not recapitulate all six here, but will give two as illustrative examples.

> *4.1 Functionality*
> *A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.*
>
> *Notes:*
>
> 1. *This set of attributes characterises what the software does to fulfill needs, whereas the other sets mainly characterise when and how it does so.*
> 2. *For the stated and implied needs in this characteristic, the note to the definition of quality applies (see 3.6).*

Since this note (3.6) will prove critical to later argumentation, we reproduce it here:

> *NOTE: In a contractual environment, needs are specified, whereas in other environments, implied needs should be identified and defined (ISO 8402:1986, note 1).*

A second quality characteristic that will be important in what follows is usability:

> *4.3 Usability*
> *A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.*
>
> *Notes:*
>
> 1. *"Users" may be interpreted as most directly meaning the users of interactive software. Users may include operators, and users and indirect users who are under the influence of or dependent on the use of the software. Usability must address all of the different user environments that the sofware may affect, which may include preparation for usage and evaluation of results.*

2. *Usability defined in this International Standard as a specific set of attributes of a software product differs from the definition from an ergonomic point of view, where other characteristics such as efficiency and effectiveness are also seen as constituents of usability.*

A key point here is that quality characteristics are the top level of an hierarchical organisation of attributes: each characteristic may be broken down into quality sub-characteristics, which may themselves be further broken down. Specific evaluations or specific views of software quality may imply that some attributes are considered to be more important than others. ISO mentions the views of the user, the developer and the manager. The manager's view is quoted here in illustration.

> *A manager may be more interested in the overall quality rather than in a specific quality characteristic, and for this reason will need to assign weights, reflecting business requirements, to the individual characteristics.*
>
> *The manager may also need to balance the quality improvement with management criteria such as schedule delay or cost overrun, because he wishes to optimise quality within limited cost, human resources and time-frame.*

The quality characteristics are accompanied by guidelines for their use. As we shall see, each attribute is associated with one or more metrics, which allow a value for that attribute to be determined for a particular system. As ISO 9126 points out:

> *Currently only a few generally accepted metrics exist for the characteristics described in this International Standard. Standards groups or organisations may establish their own evaluation process models and methods for creating and validating metrics associated with these characteristics to cover different areas of application and lifecycle stages. In those cases where appropriate metrics are unavailable and cannot be developed, verbal descriptions or "rule of thumb" may sometimes be used.*

The guidelines nonetheless suggest an evaluation process model, which breaks down into three stages.

2.1.1.1 Quality requirements definition

First comes the *quality requirements definition*, which takes as input a set of stated or implied needs, relevant technical documentation and the ISO Standard itself and produces a quality requirement specification.

2.1.1.2 Evaluation preparation

The second stage is that of *evaluation preparation*, which involves the selection of appropriate metrics, a rating level definition and the definition of assessment criteria. *Metrics*, in ISO 9126, typically give rise to quantifiable measures mapped on to scales. The *rating levels definition* determines what ranges of values on those scales count as satisfactory or unsatisfactory. Since quality refers to given needs, which vary from one evaluation to another, no general levels for rating are possible: they must be defined for each specific evaluation. Similarly, the *assessment criteria definition* involves preparing a procedure for summarising the results of the evaluation and takes, as input, management criteria specific to a particular environment which may influence the relative importance of different quality characteristics and sub-characteristics. This definition, too, is therefore specific to the particular evaluation.

2.1.1.3 Evaluation procedure

The final stage is the *evaluation procedure*, which is refined into three steps: measurement, rating and assessment. The first two are intuitively straightforward: in *measurement*, the selected metrics are applied to the software product and values on the scales of the metrics obtained. Subsequently, for each measured value, the *rating level* is determined. *Assessment* is the final step of the software evaluation process, where a set of rated levels are summarised. The result is a summary of the quality of the software product. The summarised quality is then compared with other aspects such as time and cost, and the final managerial decision is taken, based on managerial criteria.

2.1.2 The EAGLES extensions to ISO 9126

The ISO 9126 Standard sets out a general framework for designing an evaluation. The EAGLES group, together with the TEMAA LRE project (see also 3.2.1), seeks to apply this general framework to the evaluation of products in the two areas of writers' aids and translators' aids. (In earlier reports, we talked of "adequacy evaluation of market or near-market products", meaning by this that we were interested in finding ways by which someone could discover whether a particular product was adequate for their purposes.) This has led to augmentation of the ISO 9126 Standard.

The most important of these augmentations concerns the formulation of *stated or implied needs*, which, it will be remembered, are the primary input to the quality requirement definition. The EAGLES work aims at producing an evaluation package2.2.2 from which different elements can be taken and combined in different ways to reflect the needs of any particular user. There are, here, no stated needs, in the ISO 9126 sense of contractually binding specifications. What is in question are the *implied* needs of classes of users, which must be worked out through user profiling and requirements analysis techniques. So far, it has not been possible to do very much work on defining appropriate techniques for characterising users and their needs, but their importance has become increasingly obvious and is emphasised in the section on requirements analysis (2.4.4) later in this chapter.

Thinking in this way of defining the needs of classes of users and allowing the specific user to identify which of those needs fit his specific case leads to what in EAGLES is called the *consumer report paradigm* (see also 3.1). Consumer associations often publish reports on classes of products, such as dishwashers or motor cars. Individual products within each class are evaluated on a number of different dimensions and the results published in the form of a table, which gives a score for each product on each dimension. Thus, for dishwashers we might find something like:

Product name	Capacity	Programmes	Water consumption	Cleanliness	Price
XXX	12 services	6	25 litres	good	1,259
YYY	6 services	4	10 litres	average	350
ZZZ	9 services	4	15 litres	poor	965

Such tables typically do not try to make the user's choice for him: they aim to pick out characteristics of the products in question which are believed to be relevant to a user or, more plausibly, a class of users, and then to present the raw data which will allow the individual user to make an informed judgement about which product is most likely to suit his needs.

In EAGLES, we have used this idea as a way to structure our thinking about how to design an evaluation of language engineering products. We use the ISO 912 quality characteristics (A.1) as a starting point to identify attributes of products which are potentially of relevance to a class of users. We then define measures and methods whereby values for those attributes can be determined. However, we make no attempt to say in absolute terms what the relative importance of individual attributes is, or what the critical values of those attributes might be. This is determined as a function of the needs of the individual user who is a customer of the evaluation. To restate this using the terminology of ISO 9126, the quality requirements definition is based on the union of the implied needs of classes of users, appropriate metrics are selected and measurements carried out, but the individual user is left to construct his own rating level definition and assessment criteria definition.

The next section (2.2) includes a description of some preliminary attempts to provide a concrete instantiation of the EAGLES framework in the form of a parameterisable test bed:2.2.2 a software implementation which contains formal descriptions of systems or products and of characteristics of users, together with specifications of metrics and measurement methods. Parameters allow the needs of a specific user to be reflected in the choice of metrics to be applied. It should be noted though that not all metrics are automatable. In many cases, the test bed produces instructions for how a human should proceed in order to obtain a measurement for some attribute singled out as pertinent. The parameterisable test bed is a direct result of the TEMAA project.

The parameterisable test bed is of necessity based on a formal definition of evaluation2.2.1 and on formal descriptions of user characteristics and of system characteristics. In line with much current work in computational linguistics, EAGLES/TEMAA thinks in terms of features, made up of attribute/value pairs. The definition of features may come either from a consideration of the implied needs of users or from considering the characteristics of systems which already exist. Attribute/value pairs are intimately related with the metrics used to determine the values. We have already noted that whilst ISO 9126 regards metrics as ideally leading to quantifiable measures, it is recognised that this cannot always be the case. As well as metrics based on quantifiable measures, often called *tests* in EAGLES/TEMAA, EAGLES/TEMAA also recognises *facts* — attributes whose value is simply a fact such as the language dealt with by a spelling checker, and binary and scalar attributes, some of which may explicitly involve subjective human judgement. Thus attributes are typed by the kind of value they may accept. The next section (2.2) goes into more detail and gives a more formal account of the machinery we have used.

The ISO 9126 Standard quite deliberately leaves aside any discussion of how metrics are to be created or validated. Since EAGLES/TEMAA is involved in practical application of the general framework, such questions cannot be neglected. In particular, both 2.5 and the 2.6 used to obtain a measurement must be valid and reliable; that is, a metric should measure what it is supposed to measure, and should do so consistently.

The notions of validity and reliability as used within EAGLES draw on work in the social sciences and in psychology. Although there are several conceptions of validity to be found in the literature, they all essentially fall under one of two broad categories: internal (or contents) validity and external (or criteria based) validity. Internal validity is achieved by making sure that each metric adequately measures an appropriate attribute of the object to be evaluated. Internal validity is assessed by the judgement of experts. External validity is determined by calculating the coefficient of correlation between the results obtained from applying the metric and some external criterion.

A couple of examples will help to make this more concrete. Reading tests are often administered to primary school children to determine whether they can read as well as an average child of the same age. The child is required to read aloud a specially constructed text, which makes use of the vocabulary which it is considered a child of a specific age should be able to deal with. This test relies on internal validity. Whether the vocabulary chosen is or is not *well-chosen* is judged by experts on the reading skills of children.

Another test frequently administered to school children is an IQ test. The usefulness of such tests has often been the subject of contention. A frequent type of argument to be found is based on the notion of external validity: the results of the tests are shown to correlate well (or badly) with later success in academic examinations, for example, or with higher income levels in middle age.

A metric is reliable inasmuch as it constantly provides the same results when applied to the same phenomena. Reliability can be determined by calculating the coefficient of correlation between the results obtained on two occurrences of applying the metric.

Considerations of validity and reliability are not always so clear-cut as they have been made to seem in the above discussion, particularly when the evaluation is concerned with products treating a phenomenon as complex as language, and where human intervention is needed sometimes to obtain a measurement. More detailed discussion of particular metrics will raise further questions. However, the goal to be aimed at is clear.

2.2 Towards formalisation and automation

This section is based on the idea that the ultimate goal for evaluation of language engineering would be to have some automated procedure into which new products are fed which are then evaluated and compared with other products. We call the device that does this a *parameterisable test bed* (PTB).

A PTB would have several components, to be described 2.2.2. In our opinion, a major goal for future EAGLES-related work on evaluation would be to collect such components and work on the design and use of a PTB.

In this section, we will first formally define some key concepts in evaluation and then briefly describe the components of a PTB.

2.2.1 Key concepts in evaluation — a sketch for a formalisation

In the interest of standardisation, this section describes initial steps towards formalisation of evaluation.

To *evaluate* is to determine what something is worth to somebody. We describe evaluation as a *function* relating *objects* and *users* to something we will call *utility*. Utilities can sometimes be expressed in financial terms, but that does not concern us here. The important thing is that utilities represent a consistent preference relation among the items utilities are assigned to (cf. utility theory).

We then look at the nature of the *descriptions* of objects, found in e.g. consumer reports. Some useful primitives are introduced. The formal machinery is taken from the world of feature structures, well-known to computational linguists.

We then define some notions relevant to evaluation in terms of these primitives.

2.2.1.1 The evaluation function

To *evaluate* is to determine what something is worth to somebody. We can summarise this in the following function: $O \times U \to V$ where

- O is a set of objects
- U is a set of users
- V is a set of values

V: represents the idea of *utility* that drives any evaluation: the basic idea is that evaluation expresses what some object is worth to somebody; V expresses 'worth'. Utility may sometimes be related to money, but this cannot in general be assumed. We will tentatively define V as linearly ordered. This means that we can at least define relative utility by mapping object–user pairs to V.

O: represents *objects* of evaluation. Anything can in principle be evaluated, including dishwashing machines, project proposals, progress in ongoing work and evaluation procedures. In this report, we restrict O to computer programs containing some linguistic knowledge and will take all examples from this domain. The object of evaluation can be structured, i.e. it can sometimes be seen as a structure of components or functionalities that can serve as objects of evaluation themselves. For example, a translator's workbench can contain components such as a special editor, an on-line terminology database and a translation memory; the latter can be further subdivided into update and application functions. An evaluation-related question about the package as a whole may, for example, examine its integratedness or the requirements it imposes on the hardware platform. Other questions pertain to components; for example, the update/maintenance properties of a term bank may be very different from those of a translation memory.

U: represents *users*, i.e. people or organisations (potentially) interested in members of O. The notion of user is philosophically complicated. Perhaps the best view is to see it as a certain *desire*. A user is somebody who *wants* to have something or get something done. Users come in kinds. For example, the owner of a translation bureau may have a different perspective from a translator they employ. The latter may find aspects of 'user-friendliness' of some computer tool more important than the former. As a more specific example, the presence of a component such as shared terminology validation procedures in a translators' package will be more relevant for translation organisations than for freelances who work on their own.

Carrying this line of thinking further, all the factors which are often called environmental or situational variables help to define the user's desires, and are therefore part of U. If we are considering a system which can be broken down into distinguishable components, some of which may be subject to individual evaluation, we can even go so far as to say that the constraints one component of the system imposes on another (for example in the form of required output) form part of the user's desires: the user wants a task to be performed, and therefore wants all the sub-tasks of that task to be performed. Thus U includes not only all the constraints and desires resulting from the user's environment, but also, where relevant, the constraints imposed by sub-components of an overall system which might fulfill the user's needs.

We should also keep in mind that all relevant distinctions in the contexts of use can be seen as distinctions amongst types of user. In future work, U may be broken down to reflect the granularity of these distinctions. It should then become possible to see U itself as a function of conditions such as a particular kind of writer population, some specific bias in spelling errors, the fact that a personal computer has to be used as the hardware platform, etc.

The basic function given above can be curried in two ways, obtaining two perspectives on evaluation:

1. $$O \rightarrow (U \rightarrow V)$$

   describes the 'object-based' picture: given some object, evaluation tells us *who likes it*;

2. $$U \rightarrow (O \rightarrow V)$$

   gives the 'user-based' picture: given some user, evaluation tells us *what they like.*

2.2.1.2 Feature descriptions

A central role in practical evaluation is played by descriptions of objects (and components thereof) in ways that help to determine their utility value for various kinds of users. We think it is attractive to describe objects of evaluation in terms of typed feature structures, i.e. pairings of a type and an attribute-value structure.

An *object type* corresponds to a class of objects in O defined by the fact that some specific function is executed by all. Some possible object types in our domain are: editor, term bank, spelling checker.

A program will usually be indicated by a concrete or agentive noun, e.g. parser. The function it performs is usually indicated by a nomen actionis, e.g. alignment. We will use types indiscriminately to denote both the programs and the functions they fulfill.

An *attribute* refers to a property to which can be assigned, for some given member of O, one of a range of values. For example, some C compiler can be described by attributes like speed, version of the language, various debugging options, etc. Some car can be described by attributes like speed, fuel consumption, various attributes related to safety, etc.

Examples of attributes for a translation memory are:

- Its possible settings for 'fuzzy match';
- The degree to which it can automatically be expanded on the basis of a given text pair.

Attributes are typed according to their possible *values*. The range of values (scale) can for example be Boolean (yes/no), nominal or classificatory (a set of unordered values), comparative (a set of ordered values), ordinal (a range of values whose differences can be compared) or metric (real valued with fixed origin and unit). Some attributes can have other feature structures as values.

The attributes should be chosen with a view to their relevance to utility. Some general principles guiding the selection of evaluative attributes are summarised in section 2.1.1. Some further principles for attributes are described in Appendices D and E.

Furthermore, the attributes should be chosen in such a way that it is possible to establish, for a given object, what values it takes. That is, the attributes should be *measurable*. As we will see later, attributes can be typed according to the method of measurement needed to establish the value for a given object (section 2.6, on test types).

The matrices one typically sees in consumer-oriented evaluations of products are attribute-value matrices in this sense. In Appendix E, section E.3, worked-out descriptions in this form are given.

Explicit reference to user types is not commonly made in consumer reports. Sometimes, in the accompanying text, specific user profiles are mentioned (e.g. reference to people who wear spectacles in reports on binoculars); but most often, these reports assume that users know what they want. References to V are made unsystematically, e.g. in the form of a remark about some price being too high. The emphasis in consumer reports is essentially on *describing* the objects of evaluation. The reason for this is, partly, that differentiation according to user profiles is hard and the function to V often impossible to specify; and partly, that these consumer reports are typically about well-known classes of objects like dishwashers, so that the readers can be assumed to know well what they like. This latter situation does not apply to all linguistic software.

2.2.1.3 Some useful terms

Members of O perform *functions*, functions are characterised by *types* and *attributes*, attributes take *values*. Splitting up a software item in terms of a *typed feature structure* will be called *featurisation* in this section. Each type of feature can allow recursive refinement into subfeatures.

A *feature* is an attribute-value pair. Examples of features of a translation memory are the presence or absence of certain components or functions; or the values of metric attributes like a certain maximum size or speed of retrieval. A higher level feature can be complex valued (i.e. have a feature structure as value).

A *feature checklist* is like a featurisation, but the values are left open. Elaborate examples of feature checklists are given in section E.3.

A *quality characteristic* is a projection of a feature checklist (picking out a certain subset of the feature structure). Example: the basic characteristics listed in Appendix section 2.1.1.

A *specification* is a constraint on featurisations. Example: a desired dictionary can be specified as containing at least 200,000 lemmas.

Criterion, as the word is used in ordinary language, can be defined as synonymous to specification (though the pragmatics of the two words are different). In this section, *criterion* is usually used as a synonym to attribute.

A *norm* is, again, a constraint on featurisations. Norms can be used in a prescriptive way (which makes the word very similar in pragmatic meaning to specification) or in a descriptive way (describing the state of the art).

A *user profile* is a function from the domain of featurisations to V (i.e. it defines some class of users in terms of what they like or require). In Appendix E, section E.1, classes of users are exemplified.

A *test* is an object that produces values for a given attribute, given members of O. Tests can be typed by attributes tested, inputs, outputs, tools, procedures, personnel, duration, ... For a discussion of test types, see section 2.6.

2.2.1.4 Candidates for standardisation

Not all facets of the evaluation function can be standardised. Especially the relation between measurement and utility is often difficult to define explicitly. This will be called the problem of *validity*: are the attributes really related to utility? Attempts to verify the validity of featurisations in practice include estimation of effects on productivity of tools in industry, and market surveys asking whether some class of potential customers would be willing to spend a certain sum on a hypothetical product according to some specification.

Efforts towards standardisation of evaluation in the domain of natural language processing should be directed towards featurisation in the first place. The featurisations themselves cannot be standardised, but one can aim at standard feature checklists (i.e. the attributes only) per object type.

Other priority areas for investigation are user profiles and test types.

2.2.2 Parameterisable test bed

> *(...) on the construction of a PTB as a concrete instantiation of the notion of evaluation in language engineering. It should be noted that the conceptual framework developed here in the context of language engineering applies much more generally to evaluation in general.*

As said above, we feel that future EAGLES work should focus, amongst other things, on the construction of a PTB as a concrete instantiation of the notion of evaluation in language engineering. It should be noted that the conceptual framework developed here in the context of language engineering applies much more generally to evaluation in general.

It should be clear that this is a partial approach to evaluation. It needs to be supplemented with work on the relation between users' activities and goals (e.g. in working environments) and users' needs, and it should be integrated with existing views on evaluation procedures as given in ISO 9126.

A (parameterisable) test bed can only be partially automated, since many of the tests needed to evaluate objects can only be partially automated; many tests that are needed to evaluate objects are still not well understood and other tests cannot be automated in the current state of technology. We do not have computational procedures yet that can tell us which of two possible icons is more easily guessed as meaning 'cut highlighted text'. Nevertheless, we believe that the idea of an automated procedure is very important for a systematic approach to evaluation.

2.2.2.1 Parameterisable Test Bed

A Parameterisable Test Bed (PTB) is a program that:

- Is fed with parameters describing
  - Objects of evaluation;
  - Users of these objects (in a broad sense).
- Consults a library of test methods;
- Carries out the relevant tests; and
- Produces an evaluation report.

Examples of relevant objects are: spelling, grammar, style checkers; information retrieval; and translation systems. Note that a PTB may not be able to perform the whole procedure without human assistance; see below, under 'library' (2.2.2.1.3).

2.2.2.1.1 Parameters of objects Objects are described in terms of attributes and values. The set of objects is structured (subtypes, components) as well as the set of attributes (e.g. functionality-related, usability-related, etc).

2.2.2.1.2 Parameters of users Under the current view, a user is seen essentially as a list of desirables of an object. That is, a user is described in terms of the choice of attributes, weightings amongst attributes and specifications constraining attribute-values. The section on user requirements outlines some preliminary ideas on how to relate the choice of pertinent attributes systematically to the interests of users of various kinds in interaction with their goals.

2.2.2.1.3 Library The library contains for each attribute a test to compute its value. Importantly, attributes differ as to the types of test they will allow. Some attributes can be tested automatically (e.g. in the TEMAA project Thompson (1994) an automatic procedure to test functionality of spelling checkers has been developed); but for other attributes, like those related to 'user-friendliness', automated test procedures are much less feasible right now. In the latter type of case, all the PTB can do is print a report on the appropriate testing procedure. Both for automated tests and for those that rely on human co-operation, a PTB should be able to support the creation, integration and maintenance of a library of test materials, for example test suites, used by the different tests.

2.2.2.1.4 Testing The PTB will, for each relevant test, either perform it or ask the human PTB user to do this (and feed the result back into the PTB). The result is a description of each object tested in terms of attribute-values.

2.2.2.1.5 Assessment Comparison between the testing results and the user description (in terms of weighted attributes and specifications, see above, 'users' (2.2.2.1.2)) will lead to a partial ranking of objects for a given user type.

2.2.2.2 Towards a PTB for language engineering

In the TEMAA project (LRE 62–070), a very limited instance of a PTB has been implemented, which tests spelling checker functionality.

Creation of a full PTB for language engineering is an infinitely large job, since new object types may appear as well as new kinds of users. Even if restricted to a temporary state of the art PTB, creation is a huge job. However, planning in this direction is a useful thing for evaluation in language engineering in two ways:

1. The PTB can serve to organise ideas and direct researchers to standardisation of their views and methods;
2. Even very partial PTBs can already be used in practice.

The general orientation should be towards a set of programs to be used, not by end users or customers, but rather by specialised 'evaluation agents' like language-related organisations such as ELSNET or ELRA, and periodicals such as *Linguistic Industry Monitor* or various computer journals. It might even be considered that one should aim for a specialised evaluation agency for language engineering.

Here are some potential tasks for PTB creation in the near future:

- Define user categories in terms of their needs, i.e., enlarge the collection of user types known to the system; make user descriptions as independent of system descriptions as possible;
- Similarly for object types: add new ones (e.g. translation memory; grammar and style checker), refine feature checklists for those already present;
- Formalise and automate more test methods, extending the library.

As said above, evaluation for language engineering is not fundamentally different from evaluation in general. Our thinking has been influenced heavily by evaluations in other areas (cf. the 'consumer report paradigm'). Although we would like to see the creation of a PTB for language engineering, we hope also that others working on evaluation and on evaluation design may find the EAGLES work which led to the proposal of such a device fruitful and provocative of ideas in their own work, quite independently of any concrete instantiation of those ideas.

2.3 Applying the framework

Having established a general framework for evaluation, inspired by ISO 9126, and intended to be embodied by a PTB, the question arises how to set up an evaluation according to this framework or to construct a PTB to be applied to a specific class of NLP systems.

A number of ingredients is essential to fulfill this task and we will describe these briefly here.

2.3.1 Set of attributes

The central aim of the preparation stage of the evaluation is arriving at a *set of attributes* which satisfy the following conditions:

- Their values should be obtainable by observation, by direct measurement or by deriving them from the values of other attributes;

- The attributes should be adequate for expressing all explicit or implicit user requirements;
- The attributes should be general enough to be applied to sets of systems with similar tasks, and to different classes of users or usages of those systems.

The framework as such does not offer an explicit methodology for the creation of such sets of attributes other than that it commits itself to taking ISO 9126 as a starting point. However, the PTB provides us with an increasingly powerful instrument for this process, in that it contains growing libraries of attributes and their definitions, linked to validated measures and methods.

The criteria listed above will ensure a considerable amount of reusability of the various elements of the PTB and, even where those elements may not be directly reusable, the PTB will still provide us with a repertoire of options and information to guide decisions about those options.

Section 2.5 will present guidelines and exemplification on the basis of work done in the TEMAA project.

2.3.2 Requirements

The actual *requirements* (stated or implied) play a central role. These will eventually be expressed in terms of attributes. It is not always a straightforward task to extract the requirements from the user. For this problem, we have sought inspiration from the area of software requirements analysis, but it should be kept in mind that there are differences, in that the end result of such an analysis should be based on 'implementable primitives', whereas for evaluation purposes we aim at 'measurable primitives'. The process of arriving at properly expressed requirements will be iterative (or at least conceptually recursive). The PTB supports this process by allowing the cumulative reuse of components and skeletons from previous evaluations. Section 2.4 will address this problem.

2.3.3 Methods

The last main ingredient is a collection of *methods* to establish the values of the various attributes. Various questions will have to be addressed here, such as:

- How do we ensure that the values obtained satisfy validity and reliability conditions?
- What is the most appropriate setting for carrying out specific tests?

Section 2.6 will give an overview of the most commonly used testing methods in the field and will provide guidelines for identifying the most appropriate way of obtaining values for attributes.

2.4 Requirements analysis for language engineering evaluation

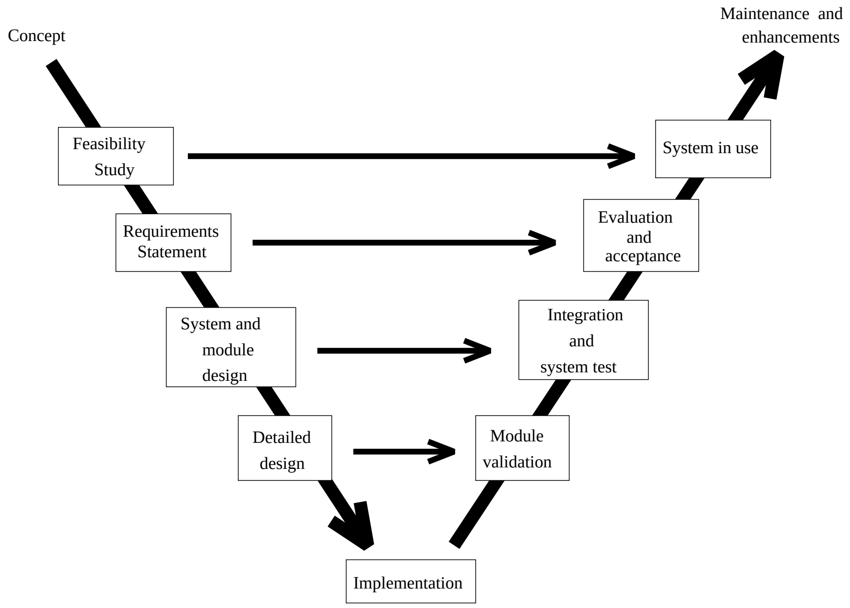
The input to the ISO process model of evaluation (2.1.1) is a statement of *stated or implied needs*. This section outlines what this should involve for language engineering products. Further detail of the approach is provided in Appendix C.

2.4.1 Requirements analysis and evaluation

First, some context is given for the place of requirements analysis in evaluation.

2.4.1.1 Evaluation in the software life cycle

In most accounts of the software engineering life cycle, evaluation is carried out in terms of the *requirements* elaborated in the first phase of the software development process: *requirements analysis*. Requirements are taken here to be equivalent to 'stated or implied needs'. More detailed, more committed descriptions of the system, which fall under the heading of design, are used to test software modules. Figure 2.1 shows how the outputs of the various design stages are typically fed into the evaluation process.



(adapted from Tansley and Hayball (1993))

Figure 2.1: 'V-diagram' showing the place of evaluation in software development.

Requirements engineering is a growing field of enquiry in software engineering and it is intended to use the products of research in requirements engineering and in the general software engineering field in this work insofar as they are useful. However, language engineering as an application area, and the particular purposes of the different kinds of evaluation distinguished by EAGLES, have special characteristics and require the development of special-purpose versions of requirements procedures.

2.4.1.2 Scope of requirements statements

The different levels of analysis and design can be thought of as different *descriptions* of a problem and its possible solutions.

The relevant space of descriptions has been described (Jackson, 1995) as covering two intersecting sets of attributes. The set $\{Di\}$ is a set of attributes of the *problem domain*, stated in terms of that domain without reference to system design decisions; the requirements statement proper is a set of relations on D or constraints involving terms from D.

The set M is a set of *machine* attributes; these are expressed in terms of entities which are parts of the actual system and not accessible to users of the system. The intersection of D and M, called S, is, in Jackson's terms, the area of *specifications*, where attributes exist in both the domain and the system design and where constraints derived from domain requirements are expressed in terms of machine requirements. In terms of the consumer report paradigm of our framework, specifications are constraints on the values of reportable attributes. Figure 2.2 illustrates the intersection.

For instance, a problem level requirement of spelling checkers for some users might be that customisations should be readily sharable between end-users. At the specification level, this might be

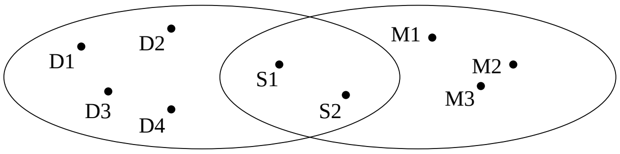


Figure 2.2: Domain and machine attributes and specifications.

expressed as the requirement that a personal dictionary must be batch copyable — that is, that there should be commands to perform the task of copying a user dictionary. Specifications can typically be thought of as relating to system functions that are directly accessible to the user. At the machine level, this will be expressed by constraints on a number of different software modules which are not directly accessible to the user.

Restricting the requirements statement to deal purely in terms of problem domain attributes, without prior commitment to any particular system's means of solving the problem, is particularly useful as a starting point for an evaluation which will be applicable to multiple software systems. This is typically the case for adequacy evaluations according to the consumer report paradigm (2.1.2), since it forms a baseline against which different ways of fulfilling these requirements might be compared and evaluated. However, there remains the problem of how to translate these domain-level requirements into a form in which it is possible to test them against all the relevant systems under test (i.e., to transform them into specifications, in Jackson's terms, or reportable attributes, in our terms). This involves not just top-down development of requirements, but is affected by the nature of the systems under test.

A reportable attribute called something like 'sharability of user dictionaries' might have a nominal value for each of the ways that existing or envisagable systems satisfy the requirement, including values corresponding to various kinds of failure to satisfy requirement at a given level.

2.4.1.3 Functional and non-functional requirements

In general, requirements are partitioned into *functional requirements* and *non-functional requirements*. Functional requirements are associated with specific functions, tasks or behaviours the system must support, while non-functional requirements are constraints on various attributes of these functions or tasks. In terms of the ISO quality characteristics for evaluation (2.1.1), the functional requirements address the quality characteristic of functionality while the other quality characteristics are concerned with various kinds of non-functional requirements. Because non-functional requirements tend to be stated in terms of *constraints* on the results of tasks which are given as functional requirements (e.g., constraints on the speed or efficiency of a given task), a task-based functional requirements statement is a useful skeleton upon which to construct a complete requirements statement. That is the approach taken in this work. It can be helpful to think of non-functional requirements as adverbially related to tasks or functional requirements: how fast, how efficiently, how safely, etc., is a particular task carried out by a particular system?

2.4.1.4 User profiling and requirements analysis

User profiles behave like *parameterisations* of requirements statements, capturing regular variation in requirements for similar types of system.

Different users may have different functional requirements, and so require different subsets of functionality to be evaluated, or they may have different non-functional constraints on functions.

The terms *user* and *user profile* will not be used so generally in what follows. It is the terminology used in our efforts towards formalisation (2.2); however, the factors that affect and parameterise software requirements do not only derive from end-users of the software, nor yet from only human elements in the software's intended environment. The term that will be used for the overall context of use is *setup*, following Galliers and Jones (1993); this will be returned to later. The actual operator of software will be called the *end-user*; the person or organisation to whom the evaluation is addressed will be called the *customer* of the evaluation.

2.4.2 Issues for requirements analysis

There is a number of issues which are important for NLP evaluation,which any requirements analysis method must take into account.

2.4.2.1 Evaluation of existing products or systems

Adequacy evaluation typically deals with the comparison of multiple, already existing software systems. The scope of the problem dealt with by each system may be different in each case, hence it may be difficult to evaluate systems against one another. Not only that, but where a set of products does appear to converge on a definition of functionality, there is no guarantee that this definition matches user and task-based requirements; these have to be established independently by the evaluator. A further complication for this type of evaluation, taking it further from the type of requirements analysis suitable for design, is that the introduction of a software system inevitably changes the tasks that preceded it.

Perhaps the most serious problem here, however, is that the needs of the user must be taken to be *implied*, since they are certainly not stated. It is here that the requirements analysis process will be most different for adequacy evaluation as opposed to progress evaluation, or any evaluation where there is not an extensionally identified user group, since realistic requirements analysis for design usually requires dialogue, prototypes and mockups to iteratively develop and test a description of the user's needs.

2.4.2.2 Reporting evaluations to the customer

The main purpose of a requirements statement from the point of view of software development is to support the subsequent design and implementation phases. When our aim is evaluation, this is no longer a priority. A new end-audience for the statement is important, however: the customer of the evaluation, the person or organisation from whose point of view the evaluation is made. If we are to take seriously the needs of the customer of the evaluation we need to analyse their requirements for information (including the form of the information) as much as the technical requirements. For the consumer report style of evaluation, the presentation will have to accommodate multiple viewpoints of this sort.

2.4.2.3 The context

It should go without saying that no meaningful adequacy evaluation can take place without a good understanding of the contexts or setups (human, organisational and computational) into which a system must fit. Ascertaining what kinds of setup are relevant to a particular evaluation, and which contextual factors affect quality requirements, is crucial to the requirements analysis process for all kinds of evaluation.

2.4.2.4 Evaluation of components and interactive systems

In the interests of software reuse and comprehensibility, a component-based approach is popular in both software engineering and LE. Various types of progress evaluation rely on the ability to devise a modular breakdown of the overall functionality such that individual components can be specified, evaluated and chosen independently, or designed, implemented and evaluated independently; however, it can be difficult to compare components against one another when they do not fit in the same way, requiring different setups in order to work at all.

For some NLP systems which provide interactive and partial support for some user task, such as grammar checkers in the task of proof-reading, it can be difficult to find a way of isolating the system's task for evaluation, because the most valid measurable results (e.g., the quality of the output text) are at a level that combines user and system performance. Analysis of the context of use, in terms of different user contributions to and constraints on performance, is then necessary.

These are in a sense the same problem, since they are about the difficulty of constructing a level ground for evaluating systems with different scopes of operation.

2.4.2.5 Evaluation of generic systems

As discussed in Galliers and Jones (1993), NLP is an area in which existing systems are seldom suitable for immediate use in a new application area. Any new use will involve more or less customisation of a generic system, for example by lexicon and grammar modification. How then can the relative suitability of different systems for a task be evaluated? This is somewhat different from the issue of evaluating component-based systems, although clearly related. It may be necessary to take into account the ease of customisation of particular generic systems, to the extent that customisability becomes a major functional requirement in its own right rather than being subsumed under the 'modifiability' quality characteristic.

2.4.2.6 Iteration and validity

The stage of requirements definition in a software project is never as self-contained as implied by the V-diagram 2.1. The V-diagram should only be taken as indicative of the relationships between the *results* of the various analysis and design, implementation and evaluation stages. The actual processes by which these results are arrived at are more likely to follow an iterative procedure, such as formalised in Boehm's spiral software development process model, (Boehm, 1988) and this is true of pure evaluation too. Ascertaining which factors are truly valid determinants of software quality is a similar problem. There needs to be a programme of validation of requirements as well as validation of how well the reportable attributes, and the measures and methods associated with them, reflect requirements. This process cannot be codified and must to a large extent be driven by open-ended evaluation in realistic situations, feeding back unforeseeable insights into the requirements statement.

2.4.2.7 Scope of evaluation

How wide should the bounds of the evaluation be set? What is the scope of the requirements? We have already seen that, when we are comparing a number of existing systems, they may not cover the same ground. When a software system is placed within a human process, as with many interactive systems, it may be relevant to evaluate the human plus software system as a whole, or at least to aim at validating and ranking requirements of software quality in terms of their correlations with the performance of the system as a whole. For instance, in the case of spelling checkers, it must be relevant to the impact of spelling checkers on the overall task of document quality assurance to know that the errors that spelling checkers are able to spot are also those that people find it easiest to spot (namely, errors that do not result in other legal but unintended words).

2.4.2.8 Knowledge acquisition for requirements analysis

New specialist research in requirements engineering is making common ground with knowledge acquisition techniques developed in Knowledge Based Systems/Expert Systems work. This addresses the problem of getting domain expertise into representations usable for subsequent design (or in our case, evaluation). NLP has its own methods, particularly for actual language analysis, but better use could be made of more general methods, particularly where whole systems and not just NLP components are being evaluated.

2.4.3 Conclusion

In these areas, it seems likely that if language engineering evaluation can make any headway, it will be contributing as much to the state of the art in requirements engineering as *vice versa*. What follows does not address all these issues adequately by any means, but they should be borne in mind for further work.

2.4.4 Requirements analysis in language engineering evaluation

Requirements analysis for evaluation aims at providing a description of the problem domain requirements. This description must support the subsequent development of detailed reportable attributes for systems under test. The requirements part proper should consist of:

- A set of *tasks* described in as much detail as possible at the problem domain level;
- A set of *scenarios* or *setups*, describing the various contexts in which the tasks should be considered and the relevant variables;
- For each task and subtask, where relevant, quality requirement definitions for functional and non-functional requirements.

(The tasks and scenarios identified can feed directly into the test design phase of evaluation, discussed in section 2.6.)

A requirements analysis procedure for language engineering evaluation aims to provide guidelines for the construction of these outputs and the final output of reportable attributes. Such guidelines will never amount to a deterministic procedure. Instead, they should form a framework which can support the activity of requirements analysis by making available:

- Libraries of task, setup and requirements elements that have been used for other evaluations or which are in some sense generic or general purpose;
- A structured set of knowledge acquisition steps and methods that can help transform generic or pre-existing tasks, setups and requirements into those required for the current evaluation.

An important part of the usefulness of generic or pre-existing requirements elements is the fact that, having been used in previous evaluations, they are associated with elements of the library of available test types.

What follows is based around the example of spelling checkers, partly to illustrate the usefulness of reuse and adaptation of pre-existing requirements definitions by comparison with the Writers' Aids subgroup work on grammar checkers (Appendix D); more detailed treatment is given in Appendix C. The work here is largely confined to the issue of adequacy evaluation.

The basic steps are:

- Establish domain-level requirements, by identifying:
  - Top level tasks at the domain level;
  - Relevant setups and their situational and environmental variables;
  - Functional requirements as specifications of top-level task input-output pairs, and non-functional requirements as constraints on aspects of top-level results.
- Transform domain-level tasks and associated setups and requirements into reportable attributes.

At each level, the potential for reuse and adaptation will be looked at.

2.4.4.1 Domain-level requirements

The first stage addresses the highest level of the requirements analysis. It is in terms of this level that other levels of analysis, down to the reportable attributes which actually express measurements of systems, must be validated.

2.4.4.1.1 Top level task description The basic tasks the system is required to address are functional requirements, which at the top level of description should relate as closely as possible to valid and measurable user requirements. For tasks which are essentially document transformation filters, such as spell checking, this is relatively straightforward, since the state of the document before the filter is a given (dependent on the setup), and the required state of the document after can usually be determined by analogy with human processes. The system under test can be evaluated, on this highest level, in terms of comparisons between two document types, the input and output to the process illustrated in Figure 2.3 .

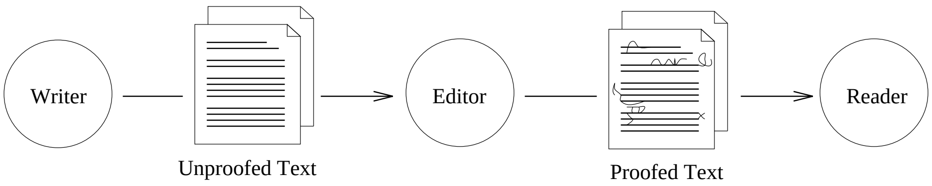


Figure 2.3: Top level data flows and agents in spell checking task.

2.4.4.1.2 Setup and variable elements The next step is to construct a set of relevant setups, identifying situational and environmental variables that affect the requirements fn the task under consideration. This includes the gathering of possibly disjoint sets of requirements from different sources, as the consumer report paradigm allows.

Questions that are relevant for the analysis of a setup include the definition of the upstream and downstream paths of information to and from the document types that form the scope of the top-level task evaluation. For example, if the text is sent to an optical character recognition device after being written, this should be noted, and a new role node inserted into the process diagram. Such nodes are used to structure the identification of variables that are relevant to task performance and facilitate the modularisation of requirements. For instance, the errors present in the text before proofing will be affected independently by variables associated with the writer role (e.g., first language and language of the text) and variables associated with the OCR role. Other relevant elements of the setup include computational and organisational settings.

Libraries of previously used setup elements, with associated quality requirements, would facilitate reuse. For instance, once the kind of spelling errors associated with OCR use have been determined, requirements stemming from these can be modularly combined with new requirements for transfer-based errors for new language pairs.

The validity of subsequent evaluation processes depends on the validity of the methods used to analyse requirements. Language engineering evaluation has special interest in the analysis of texts. Sources of representative texts need to be found; reliable experts need to analyse them; or reliable and applicable prior research must be found that characterises the relevant document types. The development of well-documented collections of representative and realistic text for a wide range of requirements is necessary.

2.4.4.1.3 Quality requirements for top level Note that this level of analysis presupposes nothing about the way the transformation or filter is to be accomplished. At this level of abstraction, we can define some quality requirements of the task at the domain level. These will form the basis of more detailed requirements at the reportable attribute level.

Functional requirements can often be defined in terms of classic recall and precision measurements: does the system do all and only what it should? For error checking systems, such requirements are based on a count of errors in the 'before' and 'after' texts. (The editor role in this process can be thought of either as a human editor in a situation before introduction of any computational tool, or as the combined role of the checking phase carried out by human and software.) Non-functional requirements at the top level might relate to the speed of the overall process of checking a document of a given sort. More detailed functional and non-functional requirements are to be found at the next level of analysis.

2.4.4.2 Transformation to reportable attributes

After this top level definition of quality, we need to turn to a consideration of the systems we are interested in evaluating. Figure 2.4 shows the place of the systems under consideration in the new task model.

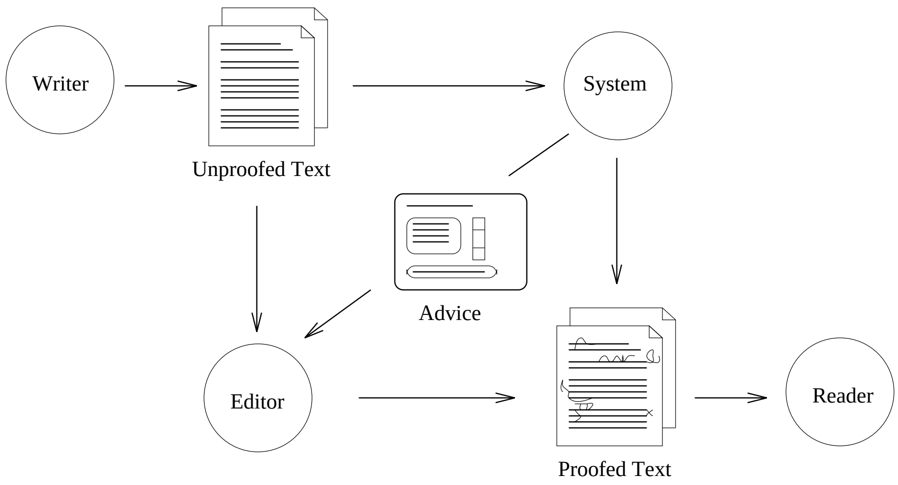


Figure 2.4: Introduction of computational system.

This task model has the basic structure of a particular sub-type of text transformation systems, namely interactive or computer-assisted text transformation systems. At this level of analysis, another set of generic quality questions becomes available. The role of the human editor, and the relations between the advice from the system and that editor, become available for analysis and the definition of quality requirements. Further knowledge acquisition is required to determine the possible variable elements in different types of human editor in terms of what kind of advice is useful. At this level of analysis too, all sorts of non-functional quality characteristics of the system become relevant, from usability to compatibility with existing software environments. Questions prompting for requirements for these quality characteristics are then associated with the task model to facilitate requirements building.

Up till now, the requirements analysis has all been top-down. To decompose the basic recall and precision functionality requirement into useful sub-attributes, we need to take a partly bottom-up approach based on the categories that are relevant to system performance. This must be based on some prior experience of the kinds of system under consideration and hence will be liable to improve from repeated open-ended evaluation. For instance, it is only because we know something about the operation and limitations of spelling checkers that we might have a separate sub-attribute for their coverage of multi-word elements like *ad hoc*.

Unlike software design, where the ostensible aim is to produce a design that is fully equivalent to the requirements, part of the purpose of evaluation is to informatively point out where designs fail to fill requirements. As an example, no spelling checkers of the normal type can correct errors stemming from simple typing mistakes which result in legal though unintended words, such as typing *form* instead of *from*; yet a realistic problem domain requirement certainly might view this as a spelling error. The discrepancy arises at the point of defining the idea of a spelling error at the attribute level as a 'best-equivalent' to its definition at the problem domain level; an explicit and structured process of decomposing or transforming problem domain requirements into measurable attributes can provide opportunities for noting where and how such discrepancies occur. While it might not be advisable to have a whole attribute devoted to reporting this failure for every system under test, such discrepancies between a problem domain requirement ('correct typos') and the available means to satisfy it should be included in any accompanying discussion or guide to the use of the attribute grid.

2.5 Measures

The requirements analysis described in the last section leads to the identification of a set of attributes whose value is potentially of interest in the context of evaluating a product or a class of products in order to assess their potential utility to a user or a class of users.

It is perhaps worth mentioning again here that, because we are interested in the needs of classes of different kinds of users, the complete set of attribute-value pairs identified may contain individual members which are mutually inconsistent or even in downright contradiction. To give a simple example, one class of users may require a constraint that any product, to be useful to them, must run on a particular hardware platform, another class that the product must not be specifically tied to that same hardware platform. Part of performing a specific evaluation is using a user profile to pick out the sub-set of attributes and critical values relevant to that particular evaluation.

2.5.1 Attributes as measures

An attribute, together with the range of values it may take, constitutes a *measure* of a system's adequacy with respect to that attribute.

For example, one attribute of spelling checkers which is very likely to be of interest is the size of the dictionary it uses. An intuitively useful range of values for that attribute is the set of integers, here representing the number of entries in the system's dictionary. The system's adequacy with respect to dictionary size is measured by the number of entries in the dictionary.

2.5.2 Validity of measures

In the section discussing ISO 9126 and our extensions to it, we have already pointed out that the validity of measures is a critical issue: if the measures used are not valid, the evaluation as a whole is worthless. Where there has been controversy about past evaluations, it is often the *validity* of the measures or of the methods used to obtain a measurement which have been called into question. Given the importance of this point, we shall recapitulate some of the earlier discussion here.

Earlier, we said that validity could be defined to be either *internal validity* or *external validity*. Let us look at these in turn.

2.5.2.1 Internal validity

We saw earlier that internal validity was achieved by making sure that what was measured reflected an appropriate attribute of the object to be evaluated. We can restate this here by saying that *internal validity* reflects the degree to which the measure directly represents the interest of the user, taking into account both the user himself and the context in which he works. An intuitively straightforward example of internal valdity in the case of spelling checkers is offered by an attribute which takes as its value the language the spelling checker deals with. It is quite hard to imagine a spelling checker for Greek being evaluated as useful for someone who had to deal with Italian text.

Ensuring internal validity is, in general, rather more difficult to achieve. It relies on the judgement of experts, in our case, the judgement of those who design the evaluation, and can only be justified after the event in the light of feedback from the customers of evaluations or from other interested outsiders.

2.5.2.2 External validity

External validity, as it was defined earlier, is achieved by demonstrating a correlation between a measure and some external criterion. It is rarely worked out formally (by, for example, actually calculating a coefficient of correlation) in the case of evaluation design, but is often used informally to justify the choice of a measure. Our earlier example of the size of dictionary used by a spelling checker affords just such an example. The size of the dictionary is of interest because, given our knowledge of how spelling checkers currently work, we believe that a small dictionary means that many false positives will be flagged, whereas a large dictionary means that (relatively) few false positives will be flagged: dictionary size correlates with signalling of false positives.

2.5.3 Reliability of measures

Measures must also be *reliable*: we call a measure reliable if there exists a reliable method for achieving the measurement. We shall return to this point later.

2.5.4 Compositional measures

Measures may be *compound*. That is, the value of an attribute may itself be structured. This is familiar from work in computational linguistics on feature structures. An example drawn from the TEMAA work on evaluating spelling checkers again concerns the lexical coverage of a spelling checker. It may well be that it is not simply the number of entries in the dictionary that matters, but the coverage of high frequency words or of some specialised technical domain, or a composite of both of these with perhaps some other factors. The value taken by an attribute reporting on lexical coverage may therefore be a composite, calculated on the basis of putting together an average or even a weighted average of the values assigned to two or more other attributes. Any one of these attributes may in its turn be a composite: there is no theoretical limit on the depth of the hierarchy of attribute-value pairs. It is clear too that, where a measure uses structured values, the validity of the measure as a whole depends on the validity of its sub-measures.

2.5.5 Typing of attributes

We have also already noted that attributes are typed by the sort of values they may accept. We list here some typical examples.

2.5.5.1 Binary values

Values may be *binary.* For example, asking whether a spelling checker allows the user to create his own personal dictionary leads to an attribute whose value is simply 'yes' or 'no'.

2.5.5.2 Classificatory values

Values may be nominative or *classificatory.* For example, we might have an attribute which asked how a new word could be added to a spelling checker dictionary. The range of possible values would be *inside checker, outside checker, not possible.*

2.5.5.3 Comparative values

Values may be *comparative.* For example, there is evidence that 'guessability' is an important factor in user acceptance of a new piece of software: if he can guess how a particular function works or what a particular icon stands for on the basis of his past experience without having to look it up, he will be happier with the software. It is hard to imagine a measure for guessability other than a simple comparative measure which awards, say, a score on a rating scale.

2.5.5.4 Numerical values

A value may also be *numerical.* For example, we might ask not only whether a user can define his own personal dictionary, but also how many personal dictionaries the spelling checker will accept.

2.5.5.5 Metric values

A value may also be a *metric.* For example, we might ask in what percentage of cases where an error is detected does the spelling checker give the correct suggestion as the first suggestion offered.

2.5.6 Conclusion

As can be seen from the above, measures also differ in the way that values for the attributes are obtained. In some cases, the value is a *fact*, which can be discovered by reading the technical literature accompanying the product, by looking at the screen or by carrying out some other mean of inspection. In other cases, the value depends on a *human judgement.* An example here is the guessability attribute mentioned above. In yet others, some sort of *test* is carried out in order to determine the value. For example, the percentage of cases in which the correct suggestion is the first offered will be determined by carrying out a test. In the interests of reliability, it is advisable to limit human intervention as much as possible: however, in the current state of the art, it is inevitable that some measures will rely on human involvement in determining the values to be assigned.

This brings us back to the general issue of reliability. In general, a measure is *reliable* if the same answer is obtained each time the same measure is applied to the same object of evaluation. This is very difficult to check when human intervention is involved. It is very important therefore that an appropriate *method* be chosen for carrying out the measurement. The next section offers some general considerations regarding methods, based on previous experience in software evaluation in general and on experience with the evaluation of language engineering systems in particular.

2.6 Methods for system measurement

The first step in any evaluation procedure should be to analyse user requirements, from which relevant quality characteristics are deduced and broken down in terms of reportable attributes. In parallel to the definition of the user, the objects of evaluation are also defined in terms of attributes. Relevant metrics are then developed, taking into account both definitions. Having defined attributes and metrics for the particular evaluation environment, the next step is to decide what methods should be applied in order to perform the measurement and to obtain values for the attributes. This section sketches the contents of a library of methods for system measurement from which the most appropriate test type and methods for data collection and data reporting can be chosen.

Among the broad software engineering community, there is little consensus concerning the exact nature of tests that are appropriate for software evaluation. Much has been said about the importance of software quality assurance from the viewpoint of project management and developers: little has been presented from the viewpoint of the end-user, who so far has played a rather passive role in software development and evaluation. Moreover, the principal aim of existing testing methods, as reported in the literature, is to uncover software errors and not to allow the adequacy of systems for a particular user environment to be assessed: naturally enough, it is this latter which is primarily of interest to users.

The methods presented here as a library for system measurement are user-oriented in the sense that evaluation is seen as being performed on behalf of a user or of a class of users. Furthermore, they are based on general software engineering principles, taking into account development-oriented testing expertise using both glass box and black box testing. (By glass box testing here we mean that access to intermediate results of processing is possible, by black box testing that only input and output behaviour can be observed.)

For ease of presentation and in order to facilitate the decision process, we make a clear distinction between the *motivation underlying testing*, which leads to a choice of the test type suitable for the particular purpose, and the *description of methods for data collection*, i.e. the instruments that can be used in combination with the test types.

This section contains only very brief descriptions of possible methods for system measurement.

2.6.1 Test types

Considering both methodological attempts to define software evaluation and practical test reports in the broad software engineering area, one may roughly distinguish three principal motivations behind testing:

1. To assess the appropriateness of a piece of software for every-day work;
2. To examine the behaviour of software under specific conditions;
3. To check the actual functionality of a piece of software.

While both software developers and users may share the principal motivation behind testing to a certain extent, the actual testing procedures will differ largely. Despite this and despite the lack of terminological consensus among leading software engineers, in order to relate the extensive testing experience of software engineers to user-oriented evaluation, we now introduce some new user-oriented testing terminology. We shall distinguish three major test types:

1. Scenario tests;
2. Systematic tests;

3. Feature inspection.

Each of these will be discussed in turn below.

2.6.1.1 Scenario tests

The term *scenario* entered software evaluation in the early 1990s. A *scenario test* is a test which aims at using a realistic user background for the evaluation of software. It is an instance of black box testing where the major objective is to assess the suitability of a software product for everyday routines. Briefly, it involves putting the system to its intended use by its envisaged type of user, performing a standardised task. Of all test types, it is the scenario test which is best suited to providing detailed empirical information on those attributes that make up the usability quality characteristic. Apart from understandability, learnability and operability, which are subcharacteristics of usability, scenario tests can also provide information on suitability, accuracy, interoperability, time behaviour, resource behaviour, changeability and adaptability.

Two different ways of performing scenario tests are reported in the software engineering literature: field tests and laboratory tests. These involve different testing environments, tasks, requirements of test systems, user participation, instruments, testing expertise and, last but not least, time and money constraints.

2.6.1.1.1 Field tests A *field test* is a type of scenario test in which the testing environment is the normal working place of the user, who is observed by one or more evaluators taking notes, recording times, etc. An obvious advantage of field tests, as compared to laboratory tests, is that the test task can include problems of data transfer between the test-system and existing systems. If the test task is to be treated as part of the daily organisational routine, the software system undergoing testing needs to be in a highly operable condition. The instruments commonly used in field tests range from the simple observation of users and noting their behaviour and interaction times on evaluation checklists, to pre- and post-testing interviews, think-aloud protocols, and, last but not least, logfile recording. The choice of instruments depends on a variety of factors such as time and money constraints, technical facilities, evaluation expertise etc.

There is an obvious connection between field testing and adequacy evaluation.

2.6.1.1.2 Laboratory tests A *laboratory test* is an instance of a scenario test in which the testing environment includes a number of isolated users who perform a given task in a test laboratory, which offers a great variety of data collection techniques. Since there is a much greater flexibility in the definition of the test task, laboratory tests are particularly useful if the system under testing is not fully operable. The artificial environment in laboratory tests allows the usage of a great number of technical instruments. Well-equipped laboratories offer one-way mirrors, video and audio recording facilities, as well as different logging programs. Whether a laboratory test is useful in the context of adequacy evaluation depends on the extent to which the tasks performed and the metrics used are revelatory of information pertinent to the user's real life needs.

The estimated costs of laboratory tests are reported to be around four times greater than those of comparable field tests. The major factor in this calculation is the very expensive maintenance of a laboratory with its various technical devices.

2.6.1.2 Systematic testing

Under the term *systematic testing* all testing activities will be subsumed that examine the behaviour of software under specific conditions with particular results expected. Whereas the objectives behind scenario testing ask for the integration of users into the testing exercise, systematic tests can be performed solely by software engineers and/or user representatives. There are three objectives that are particularly relevant for user-oriented testing, which are discussed in the following:

2.6.1.2.1 Task-oriented tested *Task-oriented testing* is performed to examine whether a piece of software actually fulfils pre-defined tasks. These tasks may either be stated in the requirements specification document, as in software development projects, or may be implied by third parties as, for instance, in consumer reports. The primary quality characteristic under investigation in task-oriented testing is functionality. The testing environment is normally the working place of the evaluator and is in principle not relevant to the interpretation of results. Task-oriented testing can be carried out during the software development process at any stage of the software life-cycle as well as with any off-the-shelf software product. The costs of task-oriented testing are comparatively small and the investment in man/hours depends on the number of tasks tested. Apart from the technical environment of the evaluator (hardware and software) no extra investment in testing equipment or instruments is normally necessary.

2.6.1.2.2 Menu-oriented testing *Menu-oriented testing* is carried out to test each program feature or function in sequence. It is prominent in many glass and black box testing techniques. The software is examined in great detail; the evaluator follows every possible path of program execution, considering each individual function as it is sequentially offered in the menu bar. Thus, while in both scenario and task-oriented testing only those functions that are necessary to perform the test tasks are performed, in menu-oriented testing each function of the software is executed at least once. As with task-oriented testing, menu-oriented testing can be performed at any stage of the software life-cycle as well as with off-the-shelf products. The costs of menu-oriented testing mainly lie in the recruitment of excellent evaluation personnel, who are capable of the ad hoc generation of metrics and data.

2.6.1.2.3 Benchmark testing *Benchmark testing* examines the performance of systems. The notion of performance can be applied either to individual functions, to system modules or to the system as a whole. In the strict technical sense, a benchmark test is a measurement of system performance which cannot be affected by variables resulting from human involvement. Typical reporting instruments applied for benchmark tests are checklists that cover the quality characteristic, the benchmark, measurement technique and results. If a benchmark involves the execution of more than one function, it is useful to construct a standard way of calling the procedures for testing purposes.

2.6.1.2.4 Systematic testing and adequacy evaluation All of the types of systematic testing described here are potentially of interest in the context of adequacy evaluation. Task-oriented testing can indicate whether a product does in fact do what a user wants it to do. Menu-oriented testing can be used to ensure that little used, but perhaps nonetheless important, functionalities are not forgotten. Benchmark testing can be used to check whether a product meets a user's minimum requirements.

2.6.1.3 Feature inspection

The aim of *feature inspection* is to be able to describe the technical features of a piece of software in as much detail as possible, so as to allow comparison between systems of the same type. Feature checklists are compiled in the awareness of the possibilities individual systems offer and with the aim of demonstrating the differences between similar tools. The results of feature inspection are meant to help consumers decide which of the systems on the market are most appropriate for their particular environment. Any feature checklist in the context of evaluation needs to be both standardised, in the sense that it should not be constrained by particular situational variables, and open, in the sense that it can cover different approaches to a problem without being prescriptive in nature.

2.6.2 Instruments

The choice of *instruments* depends on various factors such as time and money constraints, testing experience, testing environment, etc. Also, one needs to take into account what testing instruments work well in combination. Individual testing instruments used in isolation can, by their nature, only provide limited data. However, by combining two or more testing instruments, one may allow the data to be further interpreted and thus, in some sense, enriched. For example, using a logging program will provide one kind of data, videoing the user will provide another. Interpreting one set of data thus obtained in the light of the other may well contribute information which could not be garnered from either set, interpreted independently.

There are basically two types of instrument, those used for data collection in the testing phase and those used for data reporting in the reporting phase.

2.6.2.1 Testing instruments

In the testing phase, there are two major kinds of *testing instruments*, those that ask for a manual collection of data and those that perform automatic data collection. Among the most prominent manual testing instruments are questionnaires, checklists, interviews, observations and think-aloud protocols. Most automatic test instruments are developed for specific types of application, e.g. to perform benchmark tests for translation memory programs and the like. There are many different names that denote the two major types of tool used for scenario testing:

1. *Logging programs* that time-stamp and record the user interaction with the system in files that can be printed and analysed after test completion; and
2. *Playback programs* that record user interaction and provide playback facilities for later analysis.

Data obtained from the recording of non-verbalised operations, i.e. all keystrokes and mouse activities, including incorrect inputs, provide useful information on attributes related to the usability and functionality of the software.

2.6.2.2 Reporting instruments

Successful testing and evaluation is not only a matter of the choice of metrics and the optimal combination between test type and instrument for the particular test environment, but also includes a detailed, correct and adequate reporting of the test results. Among those *reporting instruments* that are of relevance for user-oriented testing are

- Evaluation descriptions;
- Test problem reports;
- Result reports; and
- Assessment reports.

2.6.2.2.1 Evaluation descriptions *Evaluation descriptions* cover all factors that influence the overall evaluation procedure, including all details that are necessary to judge the performance of the test and to verify the interpretation of results. There are four major factors that determine the type of evaluation and the corresponding testing exercise, listed here in descending order of importance:

1. The motivation behind the evaluation;
2. The system and its parameters;
3. The evaluation environment; and
4. The quality requirements that need to be tested.

2.6.2.2.2 Test problem reports *Test problem reports* provide developers with the detailed description of problems that occur during testing. They are very important instruments that aim at the improvement of software under development. Thus, they are mostly relevant for diagnostic and progress evaluation rather than for those adequacy evaluation environments in which no feedback between evaluator and developer is planned or possible.

2.6.2.2.3 Result reports *Result reports* cover all details on the metrics applied and observations made during the different testing exercises and are normally provided as an appendix to the overall test documentation. They allow interested parties to look up the detailed results of the tests.

2.6.2.2.4 Assessment reports *Assessment reports* are top-level reporting instruments that are based on the detailed data which are provided as appendices. The testing of a software system leads to a great number of individual results that are documented mainly at the metric level. In order to gain a picture of the whole software performance, it is necessary to proceed from the specific result at the metric level, through reporting on the sub-characteristics, to a general statement at the top level of quality reporting, i.e. evaluating the system's performance in terms of its eventual functionality, reliability, usability, efficiency, maintainability and portability.

2.6.3 Test materials

In this section, we briefly summarise some of the test materials frequently used in testing, mentioning some of the advantages and disadvantages associated with each.

2.6.3.1 Test sets

Test sets, in the sense used here, are collections of naturally occurring text in electronic form. Some of the best known available are the Brown Corpus of English BNC (1995), the Trésor de la Langue Française corpus of French TLF (1995), and the bilingual (English–French) material drawn from the Canadian *Hansard* LDC (1995), the latter forming a test set of parallel texts. With the power of increased computing capacity to process and store large amounts of text, a strong interest has been developing recently in collecting test sets and in defining tools able to make use of them. (See, for example, Liberman (1989)). In line with this, the Association for Computational Linguistics has launched a Data Collection Inititaive (DCI) and the Linguistic Data Consortium is specifically concerned with collecting test sets.

The LRE project MLCC is engaged in collecting plurilingual and parallel multilingual material for the European Languages.

These efforts are all concerned with collecting what might be called general material: that is, they do not aim to reflect some particular pattern of needs or some specific set of uses to which the material may be put, but rely on what text can be found in machine readable form in large quantities. Although this does not detract from the value of such collections of linguistic data, it does raise questions about their representativeness. For example, the Hansard material is clearly representative of the English and French used in the Canadian Parliament, but is highly unlikely to be representative of the French used in technical documentation or the English used in school text books. Why this is a problem is most clear in the case of evaluations designed to test a system's ability to deal with some particular type or types of text; it will be pure serendipity if a general test set contains texts of the relevant types.

On the other hand, there is a strong sense in which every test set is representative of something, and this in its turn can lead to misleading results if the test set is mistakenly taken to be representative of, say, English or French in general.

The use of general test sets must then be approached with some caution.

2.6.3.2 Test suites

By *test suite* here, we mean sets of inputs, artificially constructed and designed to probe the system's behaviour with respect to some particular phenomenon. The main problem associated with test suites is the complexity of their construction. Even at the level of syntactic phenomena, there are problems in defining inputs which will test precisely what one wants to test, and once semantic, pragmatic or translation phenomena are taken into consideration, test suite construction becomes a very delicate matter indeed.

Furthermore, test suites can quickly become unmanageably large. A principle usually adopted is to design one input per linguistic phenomenon to be tested, in order to isolate the system's behaviour with respect to that phenomenon. However, real text rarely contains one interesting linguistic phenomenon per sentence, and much of the real interest in a system's behaviour is in looking precisely at what happens when the input contains interacting phenomena. A test suite based on constructing one input per phenomenon is already large if any serious attempt is made to cover a language exhaustively. Once interactions are to be accounted for, the problem of size becomes critical. Notice, too, that size is a problem not only in constructing the test suite but in administering it and analysing the results. These and some of the problems specifically associated with constructing test suites for evaluation of machine translation systems are discussed in King and Falkedal (1990).

All these considerations constitute a good argument for thinking of general test suites, of the sort envisaged by the Hewlett Packard group, as good candidates for collaborative development. A note of caution is in order, however, if collaborative development is taken to mean simply pooling such test suites as exist. Test suites are often designed in the context of testing a specific system. There is a danger in that case that, deliberately or inadvertently, they are attuned to that particular system, thus limiting their applicability when other systems are to be evaluated.

The LRE project TSNLP (see also 3.2.4) is engaged in drawing up and exemplifying guidelines for the construction of test suites.

2.6.3.3 Test collections

By a *test collection* we mean a set of inputs associated with a corresponding set of expected outputs. In information or document retrieval, for example, a test collection consists of a set of documents, a set of queries or topics and a set of *relevance judgements*, which identify the individual documents relevant to the individual topics or queries. Typically, these elements of the test collection are divided into *training sets* and *test sets.*

The most costly element in creating such a test collection is the creation of the relevance judgements. When a large set of documents is concerned, the effort involved is so great that means are employed to allow the evaluation to be conducted with an incomplete set. Substantial effort is also required to develop the queries and topics: for the TREC evaluations mentioned in the section on recent history below (3.3.1.2), they were designed by information analysts to present varying degrees of difficulty and to cover a wide range of subject matter.

In the MUC-3 evaluation, (3.3.1.2) the filling of the templates for the terrorism test collection required not only making relevance judgements but also determining how many relevant terrorist incidents were reported in a given document (and, therefore, how many templates to generate) and which passages contained explicit or implied information pertinent to each of these incidents. This task was carried out in a co-operative venture involving the evaluators and the conference participants. Virtually every text presented difficulties of interpretation due to vagueness, ambiguity or outright self-contradiction, or due to inadequacies of the template representation or task documentation. Therefore, as the template filling task was proceeding, the documentation containing the task and output specifications had to be refined. (For a lively discussion of this effort, see Lehnert and Sundheim (1991).)

The value of such a test collection in evaluating systems comparatively to discover their adequacy with respect to the task defined by the test collection cannot be denied. The effort and cost of constructing the test collection leads naturally to wondering how the test collection can be re-used elsewhere. To take the MUC case, first, such a collection provides an obvious and valuable source of test materials to research groups working in the domain, independently of participation in the MUC conferences. A new set of issues arises if the collection is to be used for progress evaluation. In order to obtain direct comparability of results, the collection and the evaluation metrics should remain absolutely stable. Yet freezing the collection prevents it from being as good as it might be. For example, in the MUC case again, for information retrieval, additional relevance judgments will improve it, and for information extraction, higher quality filled templates will improve it.

Although re-use of a test collection can be made for subsequent formal evaluations (usually changing the test set), a change must eventually be made, with all the time and effort thus implied. This is especially true for the MUC collections, which have concentrated on just one domain. If no change is made, the collection may at some point start to hinder progress in the field, as it tends to encourage researchers to focus on certain key problems and to ignore others, and, once a system reaches a certain level of maturity, it may encourage researchers to spend more time tweaking the system than tackling the remaining major issues. However, it is certainly true that the TREC and MUC collections are sufficiently large, challenging and well-defined to support research and development in information retrieval and information extraction for a long time, even after they are no longer being used for formal evaluation.

However, the high cost of designing and constructing test collections makes it hard to imagine their being constructed outside the evaluation guided research paradigm, where the investment implied by a number of different groups working over a considerable period at essentially the same task can be used to justify the expense involved.

2.6.3.4 Conclusion

With all the test materials we have discussed, there is a tension between constructing test material which is in some sense general and can be shared across different evaluations, and the common sense feeling that most evaluations are specific, at least in the degree that the system has been constructed to carry out some specific task and should be evaluated on its ability to do so. This is as true for evaluation methodologies as it is for test materials. Designing and carrying out an evaluation is costly both in time and in money; it would be helpful to everyone concerned if ways could be found to share methodologies. But, at the same time, each evaluation is very specific to a particular system, and, perhaps even more importantly, to a specific environment in which the system should work. Increased experience and widespread discussion of evaluation techniques as a topic worthy of consideration in its own right should lead to a better consciousness of what can be shared and what not.

Chapter 3

Related Work

3.1 Looking into the consumer report paradigm

This section provides a concise literature survey of concepts and views concerning the activities of consumer organizations. Although the EAGLES Evaluation Working Group's aim is not to actually test a number of products and publish the outcomes of the tests, it shares the consumer based approach. From an EAGLES point of view, the types testing procedures applied by consumer associations will probably be the most relevant aspect of the consumer report paradigm.

3.1.1 Basic notions

The concepts of *product quality* and *consumer* can be considered the two focal points of the consumer organizations.

3.1.1.1 Consumer and product quality

In the literature, *quality* is classified in various ways. Some authors, like Beerepoot-Sangen and Leentvaar-Leistra (1991), clearly relate this concept to the interested party. They state that the producer of a product, on the one hand, will mainly be concerned with *technical quality*, that is, the extent to which the product meets the technical specifications developed for the design of that product. Although it is implied that the design satisfies the requirements and wishes of the consumer, these need not be the actual needs of the consumer: the producer can restrict himself to defining the consumer's wishes and needs without actually consulting the target group of his product.

The *quality of use*, on the other hand, refers to the extent to which the product meets the requirements that result from the purpose for which the product is to be used. The consumer's purpose can be derived from their needs and the ways in which the consumer uses a product in daily life. From this point of view, the concept of the user of a product is very closely related to that of quality.

A different characterization of the comprehensive concept of quality is the distinction between the complementary concepts of technical quality and *economic quality* (Albrecht, cited in (Box, 1979)). Technical quality, in this case, is interpreted as the utility of a product, that is, the relation between its properties of use and the consumer's expectations of use. Based upon this definition, the economic quality is paraphrased as the ratio between the utility of a product and its price (i.e. the price–quality relation).

When comparing these two classifications, the first definition of technical quality is seen to refer mainly to criteria set by the producer, whereas the second refers mostly to criteria that are relevant to the consumer. The quality of use in the classification of Beerepoot-Sangen & Leentvaar-Leistra coincides (more or less) with the second definition of technical quality. Finally, the concept of economic quality not only considers the quality of a product from the viewpoint of the consumer, but also determines the value of the product in terms of money.

3.1.1.2 Consumer and information

The concept of *consumer* can be defined as the buyer, user and disposer of products (Kanis, 1988) . When considering the purchase of a certain product, the consumer will normally try to judge beforehand (various) aspects concerning the quality of the different brands, models and types. This judgement is based on personal knowledge, experience or inspection of a product. Also, fellow-consumers can provide useful information. This type of product information is called *personal product information* (Willenborg, 1985).

In many cases, it will not be easy for the consumer to evaluate a product because they simply do not have access to essential information concerning the characteristics and functions of a product. An external source that can provide information is the producer or supplier of a certain product. However, such *commercial product information* is mainly intended to raise the sales of the product. Therefore, the information provided is often partial and persuasive, not factual enough and incomplete. Moreover, if the data are not presented in a standardised way, the consumer will not be able to compare the information for different products (Box, 1979).

A third type of information is *neutral product information*, defined as data that are provided to the consumer regarding the technical and economic functionality of a product, including durability. These data concern product characteristics that can be measured objectively and that are relevant for consumers when considering buying a product. The product information can be used not only to judge the quality of a product, but also to compare different product alternatives and to acquaint oneself with the correct ways of using a product (Willenborg, 1985) . As opposed to commercial product information, neutral product information has an objective (and independent) character and it is supplied without any commercial purposes. An important source for this kind of information is comparative product tests, a type of consumer oriented product research performed, among others, by the consumer associations. These organizations act in the interest of consumers, the users of the products, by checking the quality level of a product and establishing and standardizing evaluation criteria used for quality control.

3.1.2 Activities and aims of consumer organizations

Comparative product testing, the main activity of *consumer organizations*, can be defined as determining if, and to what degree, features which are relevant for consumers, are found to be constituent characteristics of a representative number of brands and types of a certain product. Afterwards, the compared test results (of the different products) are published simultaneously, and the prices of the products are listed (SER, cited in Box (1979)). This definition of comparative product testing emphasizes the relation between consumer and product, or, to be more precis, it relates the consumer's needs and consumer behaviour to the features of one or more products.

Comparative product tests aim at providing objective and independent information (Box, 1979) . Objectivity is guaranteed by the consumer associations: general norms and standard procedures of measurement are used as much as possible and research is done either in-house or contracted to independent institutions. The information is to be obtained independently of producers or suppliers and only the consumers' interests matter. Consumer organizations show their objectivity by not including advertisements in their magazines and by stressing their independent and even critical attitude towards the industry (Box, 1979).

The following list (Willenborg, 1985) provides some characteristics of (the results of) comparative product tests:

- Large quantities of information: detailed description of external[1] characteristics, measurement results of non-external[2] characteristics, comparative overall judgement of products and recommendations for 'best buy';
- Survey of most products (brands, models, types) of one class of product at a certain moment;
- Limited period of validity in case of rapid alternation of models and strong fluctuation of prices: retesting is necessary;
- Comparative presentation of the descriptive and evaluative information allows the consumer to obtain a view of, and insight into, the market in a relatively easy way;
- Detailed and complex information: assimilation requires motivation and intellectual skills (low convenience);

Some drawbacks (for the consumer) regarding current practices of comparative product testing, and especially its representativeness, mentioned in the literature are:

- Not all product variants are tested (Box, 1979) ;
- Frequently, tests are obsolete by the time of publication (Box, 1979) ;
- No alternative products are evaluated (e.g. comparing electric lawn-mowers and mechanical ones or spin-driers and centrifugal machines) (Beerepoot-Sangen and Leentvaar-Leistra, 1991) ;
- Little research into the actual use of products — when such research is done, it is not always known what user group has been consulted and the consumers involved are not always representative of the actual users (Beerepoot-Sangen and Leentvaar-Leistra, 1991) .

The testing activities of the consumer organizations concern, on the one hand, product oriented and, on the other, consumer oriented research. This latter type presupposes that the consumer is directly involved, mostly by means of interviews or questionnaires. Generally, however, comparative product tests, as performed by consumer associations, consist only to a small extent of consumer research. According to Box (1979), the reason for this is that the organizations hold the opinion that consumer research yields little and that the results may vary to such a degree that it would be impossible to conclude anything from it.

Therefore, the most important object of research for the consumer organizations is the technical and economic functionality of products as a measure for the satisfaction of needs by means of consumption (Kanis, 1988) . The functionality of a product results from the interaction between the product's specifications, its user and its environment (Siderius, 1989) . Examples of environmental variables are the detergent when testing washing-machines and the floor-covering when testing vacuum cleaners.

The test results, that is, the neutral product information published by consumer organizations, can be regarded as having two important functions (Box, 1979) : providing a better view of the market supply (horizontal analysis) and a more transparent market (vertical analysis). The first function implies presenting products and brands to consumers; the second implies supplying comparative information on quality aspects of the products. In this way, neutral product information will help the consumer to select a product that fulfills their (unconscious) needs.

[1] Also called non-qualitative characteristics: characteristics that can be easily determined by means of direct observation or inquiry.

[2] The opposite of external characteristics, also called qualitative characteristics.

3.1.3 Basic requirements of comparative product testing

Regarding the whole process of product testing, it is then possible to list various criteria (Siderius, 1989):

Reliability of product testing: Degree to which repetition of the measurement produces the same/similar results. Subcriteria:

Conformity: Degree to which repetition of the measurement with different specimens of a product (with identical characteristics) produces the same/similar results (product variation);

Reproducibility: Degree to which repetition of the measurement on the same object produces the same/similar results.

Validity of product testing: Degree to which the measurement results reflect the functionality of a product as experienced by the consumer. Subcriteria:

Internal validity: Degree to which variables of use and environment are included in the research conditions;

Measurability: Degree to which the product characteristics/functions of the product can be translated into measurable quantities;

Interpretability: Degree to which the test results can be translated into different aspects of the functionality of the product.

Relevance of product testing[3] : degree to which the product information enables consumers to buy and use products that contribute as much as possible to the satisfaction of their needs. Subcriteria:

- Importance of the product tested and its characteristics;
- Degree of detail in which the test results on the product's functionality are presented;
- Value judgement of certain levels or ultimate values concerning test results on the product's functionality;
- Comprehensibility/clarity of the presentation of the product information;
- Availability of the product information.

Feasibility of product testing: Amount of resources that has to be activated and checked to do the testing and to have the required methodologies at one's disposal.

Most of these criteria are interrelated: reliability is a necessary but not sufficient condition to obtain validity; reliability and validity together are necessary but not sufficient conditions to obtain relevance.

3.1.3.1 A procedure for comparative product testing

The test procedure consists of several successive phases (Willenborg, 1985; Siderius, 1989). These different stages refer to requirements concerning the testing procedure and the methods involved, complementary to the ones listed above:

1. Selecting the product class and the products to be tested;
2. Selecting and defining the attributes;
3. Developing measuring methods (measures and measuring process);

4. Performing the measurements;
5. Analysing the measuring results;
6. Interpreting and evaluating the data.

3.1.3.1.1 Selection of product class and products The first stage is described as being related to the priorities and targets of product information research. These two aspects affect the selection process.

3.1.3.1.2 Selecting and defining the attributes The second stage tries to answer the question — which product characteristics are essential when comparing product alternatives (of one class of products)? According to Willenborg (1985), essential product characteristics are those allowing one to judge:

- The primary function of the product (characteristics such as efficiency and output);
- The correct use and maintenance of the product;
- The economic value of the product (characteristics such as expenses of use, the material, contents and, if relevant, the price per standard amount);
- External effects of purchasing and/or using the product (characteristics such as aspects concerning the environment, raw material, energy, sound production, ...);
- Whether the health and safety of the user and his environment may be endangered;
- What avenues lead to legal recourse, such as the name of the manufacturer.

Other selection criteria, mentioned by Box (1979), are: (assumed) relevance for the consumer (needs and patterns of use) and testability (objectively, in a standardized way).

With regard to product characteristics, generally two types are distinguished: *non-external* and *external characteristics*. The latter type can be established and evaluated by simply observing the product or making inquiries, whereas the former requires a much more complicated/extensive analysis. Comparative product testing is primarily involved in testing non-external characteristics.

Willenborg (1985) stresses the importance of consumer research for selecting and defining product characteristics (nevertheless he is aware of possible temporal and financial restrictions). In this respect he reproduces a working method described by Cuthbert (1979):

1. By means of questionnaires and/or interviews, inquiries can be made about the ways in which consumers actually use a product. It is important that this consumer research is representative of all users.
2. Product characteristics have to be identified that are of importance to consumers. If the population of users consists of separate groups with different usage habits, it is necessary to identify the essential product characteristics for every group.
3. It should be established how the population of users appreciates the product characteristics. According to Cuthbert, the best method to obtain this information is to combine specific product characteristics into more general, complex ones, like price, lifetime, performance, expenses of use and safety, and to have them evaluated by consumers as to their relative importance. Again, it is important to consider the degree of homogeneity of the users.

3.1.3.1.3 Developing measuring methods As to the third step, developing measuring methods, Willenborg (1985) distinguishes two types of measuring method:

1. Standard methods of measuring performance (SMMPs):

   > *Test methods that lead to results having a clear relationship to the performance of a product in practical use and that are to be used as a basis for information to consumers about the performance characteristics of the product.*

   These methods are available for a limited number of products and they result from international activities on standardization (Hellman-Tuitert and Kanis, 1983) .

2. Methods developed by consumer associations:

   > *A much more elaborate set of measuring methods has been developed and applied by various consumer associations and collected (more than 800) by the International Organization of Consumer Unions (IOCU).*

Siderius (1989) stresses the importance of choosing the right measuring level and discusses the four most important levels:

1. Nominal level (presence/absence of a product characteristic);
2. Ordinal level (e.g. ordering in terms of high, middle, low);
3. Interval level (absolute numbers, averages);
4. Ratio level (relative numbers: relation between two amounts).

According to Siderius (1989), it is indispensable to measure at least at interval level to guarantee the independence of the measuring method with respect to the other phases of the test procedure.

3.1.3.1.4 Performing measurements The fourth stage, performing the measurements, produces the test results concerning the measurable quantities of the product characteristics.

3.1.3.1.5 Analysis of measuring results During the fifth stage, the test results are interpreted as data forming part of a real user situation.

3.1.3.1.6 Interpreting and evaluating data The sixth stage coincides to some degree with the previous one, but has a more subjective aspect: the test results are translated into plus and minus signs. Willenborg (1985) does not specify how this can be done, but he stresses the importance of basing the judgements on data collected with the help of consumer research. Other variables he considers important for the interpretation of the test results are norms concerning product quality minimally wished for, average product quality and variation of the product quality.

The plus and minus signs serve as a basis for determining a *global test result* per product. The individual characteristics can have different relative weights (weighting should also be based on the results of consumer research). After having compared the total test results of the various products, the consumer organizations select the best buy.

Critical remarks to be made concerning the reliability of this weighting procedure include:

- Consumers may weight the importance of different product characteristics differently from consumer associations;

- Individual consumers may perform weightings differently;
- Specimens of the same product can have different functionalities resulting from product variation;
- Weighting and best buy recommendations are to a certain degree subjective.

3.1.3.2 Translating test results into product information

Comparative product tests can be described roughly as consisting of two successive phases: a testing phase and an information phase (Siderius, 1989) . The second phase consists of interpreting the test results as product information and therefore coincides with the last step of the test procedure discussed in

3.1.3.2.1 3.1.3.1.6.

The main activity during this second phase can be described as *evaluative transformation*: the test results at interval or ratio level are translated into data situated at the ordinal level; the ordered data are then evaluated in terms of, for instance, the annotation satisfactory, almost satisfactory or unsatisfactory. According to Siderius (1989) the consumer will assimilate data more easily when they are presented in a certain arrangement expressing a value judgement.

This transformation implies applying to the test results firstly the criteria used for evaluating the products, secondly their ultimate values and thirdly the weighting factors (see paragraph 3.1.3.1.6). These three variables are set prior to the actual product testing. The procedure of setting the criteria, ultimate values and weighting factors should not only refer to the product's functionality, but also to the consumer. As for the criteria, these are mostly derived from the functional analysis of the product. However, determination of the ultimate values and weighting factors often requires additional research of a technical or sociological nature.

3.1.3.3 Some basic requirements of product information

The following criteria can be applied to judging the product information resulting from comparative product tests (Willenborg, 1985) . The information has to:

- Be complete, that is, refer to all essential product characteristics, positive and negative, that are important when evaluating different products;
- Be factual, that is, relate to product characteristics referring to technical and economic functions that can be tested objectively;
- Enable the consumer to compare various products as to their essential characteristics and to make their choice;
- Be standardized to simplify the comparison of various products: when evaluating product characteristics, the same criteria and measuring methods have to be used for every single product and the product information has to be presented unambiguously;
- Refer, if possible, to the entire collection of products belonging to one product type available on the market;
- Be topical;
- Be presented in a structured way for easy assimilation by the reader and effortless application to the evaluation of the products.

The information phase is completed when the product information has been presented (e.g. in tables) and distributed (e.g. monthly magazines). As for the presentation of the information, Box (1979) makes the following suggestions:

- Mark product attributes (in tables) that scarcely differ among products or that have little importance as to the quality of the product;
- Put the attributes (in tables) in rows instead of the brands, to simplify the comparing activity;
- Add data on nearby service facilities, product lifetime, etc.

3.1.4 Discussion (and evaluation) of the literature

With regard to information on (the topics of) consumer reports, consumer associations and product testing, it was noticeable that the relevant literature originated from rather different fields. The works consulted for this survey, for example, describe the consumer report paradigm from the point of view of the industry (Beerepoot-Sangen and Leentvaar-Leistra, 1991) , the social sciences (Box, 1979) and consumer associations in co-operation with various government departments (Hellman-Tuitert and Kanis, 1983; Kanis, 1988; Siderius, 1989; Willenborg, 1985).

Beerepoot-Sangen and Leentvaar-Leistra (1991) describe and compare the different methods for quality control of products. They look at the quality concept mainly from the viewpoint of the producer and consequently refer to quality management. Comparative product testing as carried out by the consumer organizations is treated only superficially.

Box (1979), on the other hand, looks at the relation between comparative product testing and the consumer and discusses in particular the way in which product information (i.e. the test results) influences the consumer's purchasing behaviour.

The contributions by Hellman, Kanis, Siderius and Willenborg, all published by the consumer research institute SWOKA, constitute a series of scientific reports describing the phenomena of product information and product testing. The range of existing product information systems (the sum total of product testing and translating test results into product information) is analysed and the different research phases involved in setting up such a system are presented in a structured way. One of the reports also looks into the possibility of the consumer associations having a say in, and contributing to, product innovation.

When comparing the consulted literature, it turned out that the SWOKA reports provide the most diverse and detailed information on the subject of the consumer report paradigm. Nevertheless, the reports do not really specify actual testing procedures and, as for the relevant (basic) concepts, in many cases these are not defined explicitly and unambiguously.

A rather striking aspect of the SWOKA reports is that there are two conflicting lines of reasoning. On the one hand, the authors stress the need to perform tests objectively and effectively, implying that the actual consumers should not be involved. On the other hand, it is considered necessary to establish with the greatest precision the consumers' needs and usage patterns and to include the data in the testing procedure. Some, like Willenborg (1985), also regard it as essential that the consumer is involved directly at the stages of establishing the product characteristics and interpreting the test results.

3.1.5 Literature suggestions (extracted from SWOKA reports)

The reader interested in further detail is advised to consult Capon and Burke (1980), Cary and Sproles (1978), Cude (1980), Cuthbert (1979), Geistfield, Sproles and Badenhop (1977), Groenenveld (1984), Hüttenrauch (1977), IOCU (1977), IOCU (1985), Jackel (1989), Myers and Shocker (1981), Russo (1987), Russo (1988), Silberer and Raffé (1984), Silberer (1985), Steenkamp (1989), Thorelli (1979), Thorelli and Engledow (1980) and van Weperen (1985).

3.2 Ongoing projects

3.2.1 TEMAA — A testbed study of evaluation methodologies

The TEMAA project is an LRE project carried out by researchers and developers at Center for Sprogteknologi (Copenhagen), ISSCO (Geneva), Stichting Taaltechnologie (Utrecht), Language Technology Group (Edinburgh) and Claris Ireland (Dublin) (see also Thompson (1994)).

The users involved in this project are users of spelling checkers in English, Italian, Danish and Dutch — mother tongue users, as well as foreign language users. The classes of users, apart from the managers and end-users, which will profit from the results are: developers, vendors and funders.

The objective of the project is to provide a generally applicable framework for the evaluation of NLP products. The project also studies project evaluation to some extent, but the focus is on adequacy evaluation. The framework is expressed as a number of guidelines to be followed when setting up an evaluation of a specific type of product. These guidelines are implemented in an object-oriented environment to facilitate modelling of user requirements and system characteristics.

The result is the development of a Parameterized Test Bed (PTB). Test materials for testing the lexical coverage of spelling checkers have also been constructed. The TEMAA framework is modelled in CLIPS following an object-oriented programming (OOP) paradigm. The project has also developed an experimental tool supporting automatic spelling checker checking (ASCC) which is interfaced with CLIPS. The tool is a collection of programs in WINBATCH, C and Perl. The results of the project will be documented in reports which will hopefully be widely accepted as constituting a methodological framework for evaluation to be further elaborated. Other results include an evaluation package, covering some of the test materials necessary for the evaluation of spelling checkers for Danish, English and Italian, a program, ASCC, which automatically transforms correctly spelled words into misspelled words and measures the ability of the spelling checker to find the errors and to produce a good suggestion, and a prototype of a test modelling tool for the description of an evaluation of some NLP product, i.e. an assessment of the way in which system characteristics are mapped onto different sets of user requirements. The final report will give a good description of the system. The PTB implemented in CLIPS is an experimental prototype aiming at demonstrating the soundness of the general approach, covering a subset of the user requirements and system parameters relevant to spelling and grammar checkers. Likewise, only an interesting subset of typing and spelling errors has been implemented in ASCC. The project envisages that the results could be taken over by organisations such as ISO. As for new partner types, the project envisages developers of the product type in question and standardisation bodies or user organisations.

3.2.2 COBALT

Another LRE project, COBALT, collaborated with the TEMAA project in an attempt to evaluate the EAGLES/TEMAA framework proposals by confronting them with the evaluation methods actually used in the COBALT project. COBALT works on message routing. News articles in the financial domain are automatically analysed and classified at two levels of classification to discover whether they are of potential interest to one or other of two user communities. The evaluation methods used for the different components of the system and for the system as a whole were analysed using the EAGLES framework as a way of structuring the analysis. The exercise proved very useful to all parties and confirmed that the EAGLES way of thinking about evaluation provided a constructive approach to designing evaluation methodologies also in the context of project evaluation.

3.2.3 RENOS

Yet another LRE project, RENOS, again in the framework of collaboration with the TEMAA project, used the EAGLES interim report as input to the definition of evaluation methods for the project. The project is concerned with reducing noise in a full text retrieval system by providing a new indexing system that will enable efficient retrieval of legal corpora based on concepts rather than on words. Evaluation concerns a variety of sub-components of the system as well as the system as a whole.

Once again, using the EAGLES framework as a way to structure thinking about evaluation design proved to be very fruitful.

3.2.4 TSNLP — test suites for NLP

The TSNLP (Test Suites for Natural Language Processing) project is also an LRE project. It started in December 1993 and ended in October 1995. The partners in the project are researchers from the University of Essex, DFKI GmbH (Saarbrücken) ISSCO (Geneva) and Aérospatiale (Paris). The project is concerned with the design and use of test suites in NLP (See also Balkan, Netter, Arnold and Meijer (1994)).

The background of the project is an increasing demand for large, systematic and well-documented test suites for use in evaluation and development of linguistic applications. However, test suites which meet these demands do not exist presently. Existing test suites are characterised by lack of morphological, semantic and extragrammatical phenomena, poor systematicity in testing ill-formed constructions and the co-occurrence of different phenomena, scanty documentation and annotation, and little generality. The results of applying such test suites are sometimes hard to interpret, and reuse of them is difficult.

The project's main effort consists in the production of guidelines for test suite construction and substantial amounts of test data in three languages, i.e. English, French and German. Various tools for generating test suites automatically and for lexicon replacement have also been developed during the course of the project. The results of the project are in the public domain. The main part of the test suites produced covers core syntactic phenomena and is intended for testing systems involving general syntactic processing. The remaining part has been developed for specific applications, e.g. parsers, grammar checkers and controlled language checkers.

According to the guidelines for constructing test suites developed and used by TSNLP, a limited vocabulary should be used, syntactic and semantic ambiguities should be avoided and sentences should be short and simple. The latter can be achieved by e.g. using declarative sentences in the present tense and avoiding the use of modifiers and adjuncts. To make sure that ill-formed sentences for the phenomenon to be tested are also included in the test suite, the parameters of the phenomenon should be identified and varied systematically in order to achieve an exhaustive representation of the phenomenon.

A key issue in TSNLP has been the development of an annotation scheme for annotating test suites. The annotation scheme is intended to make test suites more general and reusable. The TSNLP annotation scheme includes information on the string of words, its length, category, functional analysis and well-formedness, etc.

The project points out the advantages of test suites over corpora. These include e.g. data which is focused on specific phenomena in isolation or controlled combinations, data which reflects systematic variations over a phenomenon, non-redundant data, negative data and annotation. However, it is stressed that the two types of resource should complement each other. Another issue is the relation between test suites and evaluation type. The project points out that test suites are not exclusively useful for diagnostic and progressive evaluation. It envisages that test suites can be useful for adequacy evaluation, if they are designed in a way that takes the frequency and relevance of tested phenomena into account. This is done by providing annotations for the relevance of a test phenomenon for a particular application type, and for its frequency and weighting for a particular system and text type. However, it has been outside the scope of TSNLP to actually provide values for these last features.

3.3 Recent History

3.3.1 Some history

We open this section with a consideration of some past evaluations in the light of the notions introduced in chapter 2. No attempt is made at producing an exhaustive review: for that the reader is referred to Falkedal (1994b) and to Galliers and Jones (1993). The first gives a critical review of a number of machine translation evaluations, the second is concerned with natural language processing in general, using many insights from the field of information retrieval which has a mature and well-established evaluation methodology. Both have valuable extensive bibliographies.

The intention behind the choice of example evaluations here is to illustrate the variety of evaluation scenarios — no judgement should be inferred as to the quality of the evaluation itself.

3.3.1.1 ALPAC

The Automatic Language Processing Advisory Committee's (ALPAC) evaluation of machine translation was one of the first evaluations (ALPAC, 1966). For a fuller account, see Falkedal (1994b) and Hutchins (1986).

In terms of the framework set out in chapter 2 the evaluation as a whole can be thought of as an adequacy evaluation, comparing machine translation to human translation on the three dimensions of speed, cost and quality. Here we shall focus on one part of the evaluation, Carroll's quality assessment experiment. This part can be thought of also as a progress evaluation, assessing how close the system was to the goal of producing translations comparable in quality to human translation.

The measure used was to take a set of translations, some produced by machine, some by human translators, and ask a group of test persons to rate the translations on two scales, one for intelligibility and one for fidelity (defined in terms of informativeness — see 3.3.1.1). The use of rating scales subsequently became widespread in evaluation of machine translation (see, for example, Nagao, Tsuji and Nakamura (1988); but also see JEIDA (1992) for a critical assessment and some more recent proposals).

The method was carefully designed. The test material was 144 sentences randomly selected from four different passages of a Russian book. Six different translations were produced for the 144 sentences, three by human translators and three by different machine translation systems. The translations were then merged randomly into six sets, with the constraint that each sentence appeared only in one translation in each set. Each set was then given to three monolingual and three bilingual test persons, all of whom had had one hour's training using a set of thirty sentences drawn from the same material as the test set. There were thirty six test persons in total.

A definition in English was given for each of the points on each of the rating scales. Thus, intelligibility was rated on a nine point scale from "perfectly clear and intelligible" to "hopelessly unintelligible". Fidelity, somewhat counter- intuitively, was defined over a ten point scale in terms of informativeness . The following is Carroll's own definition of the scale (ALPAC, 1966, Appendix 10, pages 67–68) :

> *This pertains to how informative the original version is perceived to be* **after** *the translation has been seen and studied. If the translation already conveys a great deal of information, it may be that the original can be said to be* **low** *in informativeness relative to the translation being evaluated. But if the translation conveys only a certain amount of information, it may be that the original conveys a great deal more, in which case the original is high in informativeness* **relative to the translation being evaluated**.

That this definition is somewhat counter-intuitive can be deduced from how frequently it is inaccurately reported in the literature on evaluation.

The committee reached extremely negative conclusions about what could be hoped for from machine translation systems in the short to medium term. Although the report as a whole has become ill-famed and provoked much controversy both about the possible bias of the committee's membership and about the validity of the evaluation itself, the ALPAC evaluation must be considered a pioneering effort if only because it emphasised the importance of good evaluation methodologies. In our terms, though, it is clear that the measures used involved only judgements and that the perverse definition of the fidelity scale must cast some doubt on their validity.

There have been many more recent machine translation evaluations, most frequently adequacy evaluations carried out on behalf of a potential customer. In adequacy evaluation, a great deal of effort is typically required to determine what the potential customer's needs really are. Amongst many others, a customer normally will intend to use the system to translate only certain kinds of text. This need is often reflected in the use of one or more test corpora submitted for translation. In the most common case, the potential customer will have to construct a corpus reflecting his specific needs if he decides to use one.

Before leaving the topic of adequacy evaluation, it is worth making a point which is valid for commercial natural language processing systems in general. Except at the lower range, where relatively modest products such as spelling checkers can aim at relatively exhaustive coverage of the language dealt with, it is rare to find a product which will do all and only what the customer wants. Frequently, the system will have to be modified or extended to meet specific needs. Thus, evaluation is aimed at finding out not only what the system currently does but also how easily it can be modified.

3.3.1.2 ARPA

Our next example comes again from machine translation. From 1992 onwards ARPA sponsored a series of evaluations of machine translation systems. The report here is based on O'Connell, O'Mara and White (1994). The implied needs in the ALPAC evaluations were those of someone responsible for producing translations (speed, cost, quality). In the ARPA case, the needs are those of the funding agency. The declared aim of the research programme is to "further the core technology". The funding agency therefore needs a comparative evaluation of systems based on different technologies and translating from different languages into English. The difficulty of the task was further compounded by an invitation to operational systems (commercial or otherwise) from outside the research programme to participate in the evaluation exercise. Furthermore, there were great differences in the way the systems were intended to be used. At one extreme, one system was planned as a fully-automatic batch-oriented system, at the other was a system intended more as an on-line aid to a human translator than as a translation system.

Given all these constraints, the only quality characteristic which offers any hope of comparability is functionality, and that only if it is interpreted in the widest sense to allow the output of a machine-aided human translation to be compared with the output of a fully automatic machine translation. Two attributes of functionality were picked out: comprehensibility of the output and quality of the output.

In an attempt to produce direct comparability across systems translating from different languages, the test materials in the 1992 evaluation were constructed by taking a set of English newspaper articles about financial mergers and acquisitions, and having them professionally translated into the source languages the different systems worked from. The systems then translated the translations back into English and it was these outputs that were evaluated.

The metric associated with the comprehensibility attribute took the form of a comprehension test, where mono-lingual speakers of English were presented with the outputs, with output from control processes and with the original English, as well as with a set of multiple choice questions on the content of the articles.

A first problem with the validity of this metric appeared almost at once. The wide range of competence over the same task that humans can display became one of the major issues in the ARPA series of evaluations. This surfaced in the preparation of the test material. To quote:

> *However, it is evident that any human manipulation, even professional translation, has too great a potential of modifying the content of a text, and thus there is no way to tell whether a particular result reflects the performance of a system or the competence of the original translation from English. (O'Connell et al., 1994)*

In subsequent evaluations, although the comprehension evaluation was retained as a "valuable measure of the informativeness preserved by an MT system as it translates an original foreign language text into English", the back-translation method of preparing test materials was dropped.

The metric associated with the quality attribute was based on a standard US Government metric used to grade the proficiency of human translators. A panel of professional, native speaking translators of the relevant languages was asked to carry out the grading. This metric proved to be neither valid nor reliable. The grading limits of the original metric had to be changed to take account of the nature and the proliferation of errors in the machine translation output, and it proved exceedingly difficult for the quality panel to reach a consensus. The metric was dropped in subsequent evaluations.

The three subsequent ARPA evaluations have retained the comprehensibility attribute but have replaced the quality attribute with two sub-attributes, adequacy and fluency.

The metric associated with the adequacy attribute requires literate, monolingual speakers of English to make judgements determining the degree to which the information in a professional translation can also be found in a corresponding machine translation output or a control text of the same length. The information units are fragments, delimited by syntactic constituent, and containing enough information to permit comparability.

The measure of fluency is to ask the same set of persons to determine, on a sentence by sentence basis, whether the translation reads like good English. This is done without reference to a correct translation, so the accuracy of the content does not influence the judgement.

All of these three metrics are vulnerable to criticism. A comprehension test is typically used to test a human being's intelligence. How much does the test persons' intelligence interfere with the validity of the comprehensibility metric? Along similar lines, the adequacy metric is based on comparison between the information contained in a professional translation and in a machine output. To what degree does the competence of the human translator responsible for the professional translation interfere with the validity of this metric? The fluency metric relies entirely on subjective judgements: anyone who has ever produced a linguistic example for others to comment on knows to what an extent human judgements of well-formedness and fluency can differ.

However, the particular circumstances of the ARPA evaluations give rise to problems, specific to that context but sufficiently grave as to draw attention away from the more minor problems. Briefly stated, human involvement in the production of the output to be evaluated poses great problems. For the systems involved in the ARPA evaluations, human involvement covered all of post-editing of automatic machine translation, query-based interaction with a human during the translation and actual composition of the translation by a human with aid from the machine translation system. As might be expected, the competence of the human as well as his familiarity with the tools he is using greatly affect the quality of the output and the speed with which it is obtained. Disentangling the contribution of the human and the contribution of the system has proved an impossible task.

Although the severity of this problem is an artefact of ARPA's desire to compare radically different types of systems, the general point remains: ensuring that a metric measures what it is supposed to measure and only that is both critical, and, except in the case of metrics which can be completely automated, very difficult to achieve.

Neither ALPAC nor the ARPA evaluations we have discussed have come up with a very satisfactory way of evaluating machine translation systems. It is legitimate to ask why, and also to ask whether another approach might be more promising.

The answer to why, we think, has to do with the elusive notion of quality, and with the nature of translation itself. The ARPA MT evaluations do not stand alone. There is a strong ARPA tradition of comparative evaluations: the ATIS (Boisen and Bates, 1992) evaluations concerned data base query systems with a strong emphasis on spoken language; the TREC (Harman, in press) evaluations concern text retrieval; and the MUC (MUC-3, 1991) evaluations fact extraction. Although all of these, and the MUC evaluations in particular, have stimulated discussion about evaluation techniques and methodologies, none has aroused quite the strong sense of unease that the MT evaluations produce. We suspect that this is because, in the other cases, it is possible, in one way or another, to pre-define what counts as the correct answer to the problem the system is trying to solve and to evaluate a system in terms of its capacity to reproduce that answer. In the case of text retrieval, for example, it is possible to specify which texts out of a set of texts are relevant to a particular request and to evaluate a system in terms of its ability to identify automatically all and only those texts in response to the same request. This means that it is possible to define the attributes of the functionality quality characteristic quite precisely and to create and validate appropriate metrics. People may then discuss the definition of the attributes or argue about whether a different metric might not be superior in some way, but the definitions themselves are clear and unambiguous.

In the case of translation, it is impossible even to imagine making the same move. There is no such thing as *the* correct translation and one cannot imagine artificially constructing the right answer which a machine translation system should arrive at, anymore than one can imagine grading human translations by comparing them with some single perfect translation. Thus, the functionality quality characteristic cannot be given a definition in terms of a set of outputs which are the only acceptable translations of a given set of source text inputs: translation quality cannot be defined in the abstract, therefore no more can the quality requirements of a translation software be defined in the abstract.

The way out of this dilemma, we believe, is to think again about the "stated or implied needs" input to the quality requirements definition, and see them as the needs of the users not of the translation software, but of the translations produced. Translations are used for many different purposes, ranging from gleaning enough of the content of the original to know whether to put it into the waste-paper basket or not, to establishing legislation or attempting to convey the essence of a great work of literature. If we drop the idea of trying to define some abstract notion of translation quality and set about trying to find ways of measuring whether a translation is good enough for some specific purpose, this may prove more fruitful.

Space constraints prevent any fuller discussion of the wide variety of evaluation scenarios and techniques reported in the literature on machine translation. Falkedal (1994a) and MT (1994) are recent collections of papers where many of them can be found.

3.3.1.3 The Hewlett Packard Laboratories Study

Data base query is another application of natural language processing with a long history of evaluation. Woods (1973) describes informal field testing of the LUNAR system through monitoring the treatment of 110 queries during demonstration of the system and Damerau (1980) reports more extensive field testing of TQA, a transformational grammar based front end linked to a pre-existing data base of town planning data, over a period of two years from late 1977 through 1979. Both of these were clearly adequacy evaluations, with the interesting characteristic of being executed in close collaboration with the end-user community. The emphasis on field testing of data base query systems is reflected also in recent work (Jarke, Turner, Stohr, Vassiliou, White and Michielsen, 1985; Whittaker and Walker, 1989).

In this section, though, we shall concentrate on a proposal made in the context of progress and diagnostic evaluation by a group at Hewlett Packard (Flickinger, Nerbonne, Sag and Wasow, 1987). They argue that although no evaluation tool could be developed for use with natural language processing systems in general, it should be possible and useful to develop a methodology for a single application domain (data base query) in a context where there are common assumptions.

The main quality characteristic considered relevant for evaluation of a generic system (i.e. a system not specifically tailored for use with one particular data base) is the functionality of the system. The relevant attributes are linguistic and computational: the system should be able to treat a wide range of linguistic phenomena and should be able to generate the correct data base query from the natural language input.

A *test suite* was constructed to provide data for various measures relevant to these attributes. The test suite consists of a large number of English sentences annotated by a construction type. The sentences cover a wide range of syntactic and semantic phenomena, including anaphora and intersentential dependencies. Ungrammatical examples are included. Vocabulary is limited.

The method is only described in very general terms: the sentences are processed by the system being evaluated, the data base query generated and the resulting query used to query the data base. The results provide data relevant to a number of different measures: accuracy of lexical analysis, accuracy of parsing, accuracy of domain-independent semantics, correctness of the data base query generated and correctness and appropriateness of the answer. As might be expected, most of these measures are intimately related to the theory incorporated into the system. Accuracy of parsing, for example, can only be measured against what the theory of parsing implemented defines as a correct parse. The close connection between measures and theories underlying the system is typical of diagnostic evaluation; the purpose of the exercise is to provide feedback for the research workers developing the system on where modification or extension is needed.

Because knowledge of the internal workings of the system and of its theoretical underpinnings is required, the evaluation is a *glass-box* evaluation. This is in contra-distinction to the *black-box* evaluations typical of adequacy evaluations of market products, where the manufacturer will most frequently deny access to intermediate results and will describe the underlying technology only in the vaguest terms in order to protect what he sees as his own commercial interests, with the result that the evaluator must work only with the outputs produced by particular inputs and perhaps a *tertium comparationis* in terms of a pre-defined expected output.

The work described here provoked a strong interest in the construction and use of test suites, reflected in several of the papers contained in Falkedal (1994a) and in a number of on-going research projects, such as the TSNLP project discussed in section 3.2.4.

3.3.1.4 MUC–3

For our last example, we turn to the domain of fact extraction, in a context where evaluation is used to stimulate and guide research. Since 1987, the Defense Advanced Research Projects Agency (DARPA) in the USA has sponsored a series of evaluations of message understanding systems. The task is to extract material for a structured information base from a variety of naturally occurring texts. Four conferences have taken place; the description here will concentrate on the third, MUC–3. Fuller accounts can be found in Chinchor (1991), Lehnert and Sundheim (1991), MUC-3 (1991) and Sundheim (1991).

The type of evaluation is black-box, although analysis of the results in the light of the text analysis techniques used by particular systems can give some clues about how well particular techniques fare. The basic strategy is to define a goal by creating a test collection consisting of a set of texts and a set of relevance criteria for the texts and for the information to be extracted. A set of "answer templates" is then defined, and the system evaluated by comparing the templates it produces with the answer templates. Fifteen systems participated in MUC–3. The evaluation is thus a comparative evaluation of these systems' adequacy in fulfilling the particular task. Re-testing a system after modification using the same material can also produce an evaluation of that system's progress.

Once again, the only quality characteristic taken to be relevant is the system's functionality. The attribute is the system's ability to extract essential information of the specified kind and the measure is the number of template slots correctly filled in each case.

MUC–3 used a corpus of 1,600 articles, each about half a page long drawn from a variety of text types, including newspaper articles, television and radio news, speech and interview transcripts, etc. The articles were divided into a training set of 1,300 texts made available to all fifteen participating sites and a test set of 300 articles. The articles covered a wide range of linguistic phenomena and included ungrammatical input.

The fifteen different systems included pattern matching systems, where there was fairly direct mapping from text to slot fillers, syntax-driven systems in which a traditional syntactic structure was produced and input to subsequent processing, and semantics-driven systems, guided primarily by semantic predictions but perhaps also using some degree of syntactic information and/or some pattern matching.

The specific task was to extract information on terrorist incidents, such as incident type, date, location, perpetrator, target, instrument, outcome, etc., from the 300 articles in the test set. Many of the articles were irrelevant to the task and relevant articles did not only contain relevant information.

In preparation, all participants manually generated an agreed set of filled templates from the training set according to a set of relevance criteria and rules refined during the generation process. Performance for systems on the test set of texts was evaluated by reference to an independently provided set of answer templates for the articles and information that ought to have been selected. A semi-automated scoring program was developed to calculate the various measures of performance. The two primary measures were *completeness* (analogous to *recall* in information retrieval) and *accuracy* (analogous to *precision* in information retrieval). Completeness was calculated as the ratio between the number of template slots filled correctly by the system and the total number of filled slots in the answer template. Fills corresponding exactly to the fill in the answer template scored 1.0, whilst fills judged by humans to be a good partial match scored 0.5. Accuracy was the ratio of slots correctly filled to the number of total fills generated. Two other measures, *over-generation* and *fallout*, were also used, but will not be discussed here.

The method was particularly stringent. The test could be executed only once. Systems that crashed were allowed to restart but were not allowed to re-process any message that caused a fatal error. Lehnert and Sundheim (1991) note that this procedure resulted in scores for some sites that did not reflect true system capability. They also report that despite the stringency of the method, eight sites achieved at least 20% completeness and 50% accuracy. Two systems exhibited completeness scores over 40% with accuracy over 60%.

If we look at the results in general, all systems performed better on accuracy than on completeness. The linguistically based systems could not adjust by increasing completeness at the expense of accuracy. Many participants, when information derived from sentences had to be re-organised into target template instantiations, identified discourse analysis as a major trouble spot. No participant claimed to have a satisfactory discourse component. The four top scoring systems differed very widely in their text analysis techniques. One high-ranking system worked with a 6,000 word dictionary, no formal grammar and no syntactic parse trees, and a close competitor operated with a 60,000 word dictionary, a syntactic grammar and syntactic parse trees for every sentence encountered. However, when the systems were ranked according to the highest combined accuracy and precision scores, the top eight systems were all natural language processing systems rather than systems using exclusively stochastic or inductive methods.

As already mentioned, the MUC evaluations stand in a tradition created and maintained by DARPA and ARPA of careful and rigorous evaluation techniques being used for comparative evaluation of a number of systems which may all work according to quite different principles. Although such evaluation provides clear information on a system's adequacy with respect to the particular task in hand, it is difficult to estimate its usefulness in predicting the capacity of individual systems to evolve and progress or the portability of systems trained on texts drawn from one domain to a different domain. Nonetheless, it cannot be denied that the existence of test collections such as those developed for MUC–3 constitute a valuable resource for the research community.

As noted earlier, the same philosophy of applying black-box evaluation comparatively to a number of different systems working on different principles is now also being adopted with on-going research on machine translation. The evaluation methodology, however, is rather different: with the MUC evaluations it is possible to specify the task to be accomplished quite precisely, since a critical element of the test collection is a set of correct answers. This cannot be done for machine translation; it is in the nature of translation that, for any given text, potentially many translations would all be equally acceptable.

Even the very small number of evaluations described here is enough to support the conclusion that evaluations vary enormously in their purpose, in their scope and in the nature of the object being evaluated. Consequently, it is hardly surprising that evaluation techniques in their turn differ widely, as do the resources they require. It is this observation which led Flickinger et al., as noted earlier, to the conclusion that it is in principle impossible to envisage the design and construction of some general evaluation tool, into which any natural language processing system could be plugged in order to obtain data relevant to a set of informative measures. Galliers and Jones (1993) formulate the same hypothesis rather nicely:

> *The many elements involved in evaluation, perspective and levels on the one hand and system structure and applications on the other, mean that it is quite unreasonable to look for common, or simple evaluation techniques. The Workshop ... instead emphasised the need for very careful analysis of what is involved and required for any individual evaluation, and the limited extent to which evaluation techniques can be effortlessly, routinely or even legitimately transferred from one case to another. It is essential to determine comparability before such transfers can be made, and as in many situations true comparability will be very restricted, what can be common in the field is rather a set of methodologies and an attitude of mind. Thus evaluation in NLP is best modelled by analogy with training the cook and supplying her with a good batterie de cuisine.*
>
> (Galliers and Jones, 1993, 105)

We might also ask whether it is not possible to share some of the resources used in evaluation. It is intuitively clear that test materials like test collections or test suites are expensive to produce and maintain: there would be obvious interest in producing materials which could be shared by the community as a whole and re-used in different evaluations. Furthermore, use of the same test material might be expected to produce evaluation results which could more easily be compared, thus leading to another kind of shared resource.

Chapter 4

Conclusions and Directions for Future Work

We have emphasized many times throughout this report that we make no claim to finished work: we have concentrated our work during this first thirty months on one particular kind of evaluation, adequacy evaluation of on market or near market products. We have only been able to carry out hands-on work with products in two areas, writers' aids and translators' aids, and we have only been able to do very preliminary work on the evaluation of knowledge management systems. Although we feel that substantial progress has been made towards the overall goal of defining a general framework for the design of evaluation methodologies, we are aware that much remains to be done. In particular, work during this first complete period has led to the identification of areas where very little indeed has been done, either within the group or in the larger community and which we feel should be given priority in follow-up work.

Obvious directions for future work include applying the framework to other kinds of evaluation and to other areas, perhaps even outside the strict domain of language engineering.

In addition to these obvious areas for future work, the work to date has also revealed the importance of work on user profiling and requirements analysis undertaken in the spirit of identifying and formalising the needs of classes of users. Very little has yet been done on this topic which is of importance to a very wide community, going beyond the bounds of language engineering products to touch the concerns of industry as a whole and potentially providing insights useful to research policy and research and development projects.

On a very practical level, we have been able to sketch the possibility of semi-automated evaluation test-bed which can be parametrized to take account of the characteristics of different objects of evaluation and of different kinds of users who are interested in the results of an evaluation, We think this, too, worthy of further effort and potentially of interest to both industry and the wide community of users.

The territory when we started was almost virgin: it is perhaps worthwhile to close by summarizing where we consider we have started to make inroads into it. We believe that we have achieved the following:

- Clarity of conceptual thinking about first principles of evaluation and kinds of evaluation (see 2.2.1).
- Design of a general framework for evaluation purposes (see 2) which has already proved its usefulness in a number of ways.
- Design of a methodology for establishing measures (see 2.5) and methods of measurement (see 2.6).

- Design of a methodology for the formal description of systems (see 2.2).
- Design of a methodology for the formal description of classes of users (see 2.4).
- Initial elaboration of feature checklists (see E.3.2).
- Initial testing of the methodologies and the framework (see I).
- Initial work on user profiling and a proposal for fruitful continuation of that work (see C).
- Initial elaboration of user profiles (see F).
- A substantial case study of one organisation (see K).
- Surveys of existing related standardisation activities (see A).
- Summaries of previous relevant work (see 3.3).
- Surveys of relevant products on the market (see G).
- Compilation of a large bibliography (see L).
- Stimulation of interest in the wider community in matters pertaining to evaluation, given concrete form through the establishment of some important feedback channels and collaborations (see 3.2).

It would be wrong to close this report without thanking again all those who have contributed to it, whether directly by volunteering their labour, by writing sections of it and by providing feedback and comment on this and earlier versions, or indirectly by providing additional financial and moral support. We hope the same spirit of collaboration which has informed our efforts to date will continue into the future.

Appendix A

ISO Terms and Guidelines

A.1 Quality characteristics and subcharacteristics

These are the six software quality characteristics and their definitions:

A.1.1 Functionality

A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs. (ISO 9126: 1991, 4.1)

A.1.2 Reliability

A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. (ISO 9126: 1991, 4.2)

NOTE — *Wear or ageing does not occur in software. Limitations in reliability are due to faults in requirements, design and implementation. Failures due to these faults depend on the way the software product is used and the program options selected rather than on elapsed time.* (ISO 9126: 1991, 4.2)

A.1.3 Usability

A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. (ISO 9126: 1991, 4.3)

NOTES

1. *'Users' may be interpreted as most directly meaning the users of interactive software. Users may include operators, end users and indirect users who are under the influence or dependent on the use of the software. Usability must address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.*

2. *Usability defined in this International Standard as a specific set of attributes of software product differs from the definition from an ergonomic point of view, where other characteristics such as efficiency and effectiveness are also seen as constituents of usability.* (ISO 9126: 1991, 4.3)

A.1.4 Efficiency

A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions. (ISO 9126: 1991, 4.4)

NOTE — *Resources may include other software products, hardware facilities, materials, (e.g. print paper, floppy disks) and services of operating, maintaining or sustaining staff.* (ISO 9126: 1991, 4.4)

A.1.5 Maintainability

A set of attributes that bear on the effort needed to make specified modifications. (ISO 9126: 1991, 4.5)

NOTE — *Modifications may include corrections, improvements or adaptation of software to changes in environment, and in requirements and functional specifications.* (ISO 9126: 1991, 4.5)

A.1.6 Portability

A set of attributes that bear on the ability of software to be transferred from one environment to another. (ISO 9126: 1991, 4.6)

NOTE — *The environment may include organizational, hardware or software environment.* (ISO 9126: 1991, 4.6)

A.2 Subcharacteristics

The following list defines the respective subcharacteristics:

A.2.1 Functionality

Suitability:
Attribute of software that bears on the presence and appropriateness of a set of functions for specified tasks. (ISO 9126: 1991, A.2.1.1)

Accuracy:
Attributes of software that bear on the provision of right or agreed results or effects. (ISO 9126: 1991, A.2.1.2)

Interoperability:
Attributes of software that bear on its ability to interact with specified systems. (ISO 9126: 1991, A.2.1.3)

NOTE — *Interoperability is used in place of compatibility in order to avoid possible ambiguity with replaceability.* (ISO 9126: 1991, A.2.1.3)

Compliance:
Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions. (ISO 9126: 1991, A.2.1.4)

Security:
Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data. (ISO 9126: 1991, A.2.1.5)

A.2.2 Reliability

Maturity:
Attributes of software that bear on the frequency of failure by faults in the software. (ISO 9126: 1991, A.2.2.1)

Fault tolerance:
Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface. (ISO 9126: 1991, A.2.2.2)

Recoverability:
Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it. (ISO 9126: 1991, A.2.2.3)

A.2.3 Usability

Understandability:
Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability. (ISO 9126: 1991, A.2.3.1)

Learnability:
Attributes of software that bear on the users' effort for learning its application (for example, operation control, input, output). (ISO 9126: 1991, A.2.3.2)

Operability:
Attributes of software that bear on the users' effort for operation and operation control. (ISO 9126: 1991, A.2.3.3)

A.2.4 Efficiency

Time behaviour:
Attributes of software that bear on response and processing times and on throughput rates in performing its function. (ISO 9126: 1991, A.2.4.1)

Resource behaviour:
Attributes of software that bear on the amount of resources used and the duration of such use in performing its function. (ISO 9126: 1991, A.2.4.2)

A.2.5 Maintainability

Analysability:
Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified. (ISO 9126: 1991, A.2.5.1)

Changeability:
Attributes of software that bear on the effort needed for modification, fault removal or for environmental change. (ISO 9126: 1991, A.2.5.2)

Stability:
Attributes of software that bear on the risk of unexpected effect of modifications. (ISO 9126: 1991, A.2.5.3)

Testability:
Attributes of software that bear on the effort needed for validating the modified software. (ISO 9126: 1991, A.2.5.4)

A.2.6 Portability

Adaptability:
Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered. (ISO 9126: 1991, A.2.6.1)

Installability:
Attributes of software that bear on the effort needed to install the software in a specified environment. (ISO 9126: 1991, A.2.6.2)

Conformance:
Attributes of software that make the software adhere to standards or conventions relating to portability. (ISO 9126: 1991, A.2.6.3)

Replaceability:
Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software. (ISO 9126: 1991, A.2.6.4)

NOTES

1. *Replaceability is used in place of compatibility in order to avoid possible ambiguity with interoperability.*
2. *Replaceability with a specific software does not imply that this software is replaceable with the software under consideration.*
3. *Replaceability may include attributes of both installability and adaptability. The concept has been introduced as a subcharacteristic of its own because of its importance.*

A.3 Terminology

The following list defines terms that could be relevant for evaluation purposes:

Assessment: *An action of applying specific documented assessment criteria to a specific software module, package or product for the purpose of determining acceptance or release of the software module, package or product.* (ISO 9126: 1991, 3.1)

Customer: *Ultimate consumer, user, client, beneficiary or second party.* (ISO 9004: 1987, 3.4)

Defect: *The nonfulfilment of intended usage requirements.* (ISO 8402: 1986, 3.21)

Features: *Features are identified properties of a software product which can be related to the quality characteristics.* (ISO 9126: 1991, 3.2)

Firmware: *Hardware that contains a computer program and data that cannot be changed in its user environment. The computer program and data contained in firmware are classified as software; the circuitry containing the computer program and data is classified as hardware.* (ISO 9126: 1991, 3.3)

Inspection: *Activities such as measuring, examining, testing, gauging one or more characteristics of a product or service and comparing these with specified requirements to determine conformity.* (ISO 8402: 1986, 3.14)

Level of performance: *The degree to which the needs are satisfied, represented by a specific set of values for the quality characteristics.* (ISO 9126: 1991, 3.4)

Liability (product/service): *A generic term used to describe the onus on a producer or others to make restitution for loss related to personal injury, property damage or other harm caused by a product or service.* (ISO 8402: 1986, 3.19)

Measurement: *The action of applying a software quality metric to a specific software product.* (ISO 9126: 1991, 3.5)

Nonconformity: *The nonfulfilment of specified requirements.* (ISO 8402: 1986, 3.20)

NOTE — *The basic difference between 'nonconformity' and 'defect' is that specified requirements may differ from the requirements for the intended use.* (ISO 8402: 1986, 3.20)

Quality: *The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.* (ISO 8402: 1986, 3.1)

Quality assurance: *All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.* (ISO 8402: 1986, 3.6)

Quality control: *The operational techniques and activities that are used to fulfill requirements for quality.* (ISO 8402: 1986, 3.7)

Quality surveillance: *The continuing monitoring and verification of the status of procedures, methods, conditions, processes, products and services, and analysis of records in relation to stated references to ensure that specified requirements for quality are being met.* (ISO 8402: 1986, 3.11)

Rating: *The action of mapping the measured value to the appropriate rating level. Used to determine the rating level associated with the software for a specific quality characteristic.* (ISO 9126: 1991, 3.7)

Rating level: *A range of values on a scale to allow software to be classified (rated) in accordance with the stated or implied needs. Appropriate rating levels may be associated with the different views of quality i.e. Users, Managers or Developers. These levels are called rating levels.* (ISO 9126: 1991, 3.8)

Recoverability: *Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.* (ISO 9126: 1991, A.2.2.3)

Reliability: *The ability of an item to perform a required function under stated conditions for a stated period of time. The term 'reliability' is also used as a reliability characteristic denoting a probability of success or a success ratio.* (ISO 8402: 1986, 3.18)

Replaceability: *Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.* (ISO 9126: 1991, A.2.6.4)

Resource behaviour: *Attributes of software that bear on the amount of resources used and the duration of such use in performing its function.* (ISO 9126: 1991, A.2.4.2)

Security: *Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.* (ISO 9126: 1991, A.2.1.5)

Software: *Intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a data processing system.* (ISO 9000-3: 1991, 3.1)

Software product: *Complete set of computer programs, procedures and associated documentation and data designated for delivery to a user.* (ISO 9000-3: 1991, 3.2)

Software item: *Any identifiable part of a software product at an intermediate step or at the final step of development.* (ISO 9000-3: 1991, 3.3)

Software quality: *The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs.* (ISO 9126: 1991, 3.11)

Software quality assessment criteria: *The set of defined and documented rules and conditions which are used to decide whether the total quality of a specific software product is acceptable or not. The quality is represented by the set of rated levels associated with the software product.* (ISO 9000-3: 1991, 3.12)

Software quality characteristics: *A set of attributes of a software product by which its quality is described and evaluated. A software quality characteristic may be refined into multiple levels of sub-characteristics.* (ISO 9126: 1991, 3.13)

Software quality metric: *A quantitative scale and method which can be used to determine the value a feature takes for a specific software product.* (ISO 9126: 1991, 3.14)

Specification: *The document that prescribes the requirements with which the product or service has to conform.* (ISO 8402: 1986, 3.22)

Stability: *Attributes of software that bear on the risk of unexpected effect of modifications.* (ISO 9126: 1991, A.2.5.3)

Suitability: *Attribute of software that bears on the presence and appropriateness of a set of functions for specified tasks.* (ISO 9126: 1991, A.2.1.1)

Testability: *Attributes of software that bear on the effort needed for validating the modified software.* (ISO 9126: 1991, A.2.5.4)

Time behaviour: *Attributes of software that bear on response and processing times and on throughput rates in performing its function.* (ISO 9126: 1991, A.2.4.1)

Understandability: *Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability.* (ISO 9126: 1991, A.2.3.1)

Usability: *A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.* (ISO 9126: 1991, 4.3)

Validation (for software): *The process of evaluating software to ensure compliance with specified requirements.* (ISO 9000-3: 1991, 3.7)

Verification (for software): *The process of evaluating the products of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase.* (ISO 9000-3: 1991, 3.6)

A.4 Guidelines

A.4.1 After-sales servicing

Source: ISO 9004: 1987, 16.2.3 and 16.2.4.

Instructions for use dealing with the assembly and installations, commissioning, operation, spares or parts lists, and servicing of any product should be comprehensive and supplied in a timely manner. The suitability of instructions for the intended reader should be verified.

Assurance should be provided for an adequate logistic back-up, to include technical advice, spares or parts supply, and competent servicing. Responsibility should be clearly assigned and agreed among suppliers, distributors and users.

A.4.2 Evaluation of importance

Source ISO 9004: 1987, 15.3.

The significance of a problem affecting quality should be evaluated in terms of its potential impact on such aspects as production costs, quality costs, performance, reliability, safety and customer satisfaction.

A.4.3 Evaluation process

Source ISO 9126: 1991, 5.3.

The evaluation process consists of three stages and it may be applied in every appropriate phase of the life cycle for each component of the software product:

Quality Requirement Definition: *The purpose of the initial stage is to specify requirements in terms of quality characteristics and possible subcharacteristics. Requirements express the demand of the environment for the software product under consideration, and must be defined prior to the development. As a software product is decomposed into major components, the requirements derived from the overall product may differ for the different components.*

Evaluation Preparation: *The purpose of the second stage is to prepare the basis for evaluation.* This stage consists of three components:

Quality metrics selection : *The manner in which quality characteristics have been defined does not allow their direct measurement. The need exists to establish metrics that correlate to the characteristics of the software product. Every quantifiable feature of software and every quantifiable interaction of software with its environment that correlates with a characteristic can be established as a metric. [...] Metrics can differ depending on the environment and the phases of the development process in which they are used. Metrics used in the development process should be correlated to the user respective metrics, because the metrics from the user's view are crucial.*

Rating levels definition : *Quantifiable features can be measured quantitatively using quality metrics.* The result, the measured value, must be interpreted as a rated value, i.e. *divided into ranges corresponding to the different degrees of satisfaction of the requirements. Since quality refers to given needs, no general levels for rating are possible. They must be defined for each specific evaluation.*

Assessment criteria definition : *To assess the quality of the product, the results of the evaluation of the different characteristics must be summarized. The evaluator has to prepare a procedure for this, using, for instance, decision tables or weighted averages. The procedure usually will include other aspects such as time and cost that contribute to the assessment of quality of a software product in a particular environment.*

Evaluation Procedure:

The last step of the Evaluation Process Model is refined into three steps, namely measurement, rating and assessment.

Measurement: *For measurement, the selected metrics are applied to the software product. The result is values on the scales of the metrics.*

Rating: *In the rating step, the rating level is determined for a measured value.*

Assessment: *Assessment is the final step of the software evaluation process where a set of rated levels are summarized. The result is a statement of the quality of the software product. Then the summarized quality is compared with the other aspects such as time and cost. Finally managerial decision will be made based on the managerial criteria. The result is a managerial decision on the acceptance or rejection, or on the release or no-release of the software product.*

A.4.4 Maintenance of software products

Source ISO 9000-3: 1991, 5.10.

Maintenance activities for software products are typically classified into the following:

a) problem resolution;

b) interface modification;

c) functional expansion or performance improvement.

[...]

All changes to the software (for reasons of problem resolution, interface modifications, functional expansion or performance improvement) carried out during maintenance should be made in accordance with the same procedures, as far as possible, used for the development of the software product. All such changes should also be documented in accordance with the procedures for document control and configuration management.

a) Problem resolution: *Problem resolution involves the detection, analysis and correction of software nonconformities causing operational problems. When resolving problems, temporary fixes may be used to minimize downtime and permanent modifications carried out later.*

b) Interface modifications: *Interface modifications may be required when additions or changes are made to the hardware system, or components, controlled by the software.*

c) Functional expansion or performance improvement: *Functional expansion or performance improvement of existing functions may be required by the purchaser in the maintenance stage.*

[...]

The items to be maintained, and the period of time for which they should be maintained, should be specified in the contract. The following are examples of such items:

a) program(s);

b) data and their structures;

c) specifications;

d) documents for purchaser and/or user;

e) documents for supplier's use.

A.4.5 Quality objectives

Source ISO 9004: 1987, 4.3.1.

For the corporate quality policy, management should define objectives pertaining to key elements of quality, such as fitness for use, performance, safety and reliability.

A.4.6 Quality system elements

Source ISO 9004: 1987, 0.4.

Risk considerations: *consideration has to be given to risks such as those pertaining to the health and safety of people, dissatisfaction with goods and services, availability, marketing claims and loss of confidence.*

Cost considerations: *consideration has to be given to safety, acquisition cost, operating, maintenance, downtime and repair costs, and possible disposal costs.*

Benefit considerations: *consideration has to be given to reduced costs, improved fitness for use, increased satisfaction and growth in confidence.* (ISO 9004: 1987, 0.4)

Appendix B

Methods for System Measurement

B.1 Preface

This appendix is intended as input to the existing EAGLES framework for evaluation. So far the main effort of the EAGLES evaluation working group has been directed towards establishing a framework for evaluation that distinguishes the type of evaluation (adequacy, progress or diagnostic evaluation), and outlines the construction of a quality requirement definition (see (ISO, 1991)) based on user profiling and system featurisation, giving as concrete examples a selection of detailed metrics for some attributes of writers' and translators' aids.

B.2 Introduction to the testing of software

Even among the broad software engineering community, there is little consensus concerning the exact nature of tests that are appropriate for software evaluation. Much has been said on the importance of software quality assurance from the viewpoint of project management and developers. Little has been presented as seen from the viewpoint of the end-user who so far has occupied a rather passive role in software development and evaluation. However, whereas software testing until recently was the sole domain of software developers, a gradual change in the underlying philosophy can now be perceived, giving the user a higher priority in both the overall software development process and as a customer. The fact that software developers are not necessarily the best choice when it comes to testing their own programming product is gaining general acknowledgement (Thaller, 1994, 61,84,90), cf. also (Hausen and Müllerburg, 1982, 119).

Moreover, a new philosophy has emerged concerning the goals of testing: whereas so far the principal motivation for applying software tests was "... to look actively for problems so that these may be corrected thus improving the software" (Bukowski, 1987, 369), cf. also (Sneed, 1987, 10.3–2), the evaluation and comparison of *existing* software products is gaining more and more importance. This is due to the fact that the PC market is overloaded with competing software packages for different application areas and users are at a loss when it comes to deciding which of the products suits their purpose best. The type of information which is needed in this situation is very different from what existing testing exercises can provide. Given this rather new user-centered development philosophy, it is obvious that there is a lack of methodologies for user-centered evaluation and testing. This is even more true for the NLP area which has so far produced only a relatively small number of operable systems.

This report strives to develop a user-oriented testing framework which is based on general software engineering (SE) principles, considers development-oriented testing expertise of both glass box and black box testing, and, last but not least, applies and further develops testing approaches to the needs of user-oriented assessment, particularly for NLP applications. Despite the great variance in terminology and the differences of the actual testing environments, common goals and procedures in testing will be identified, considering elementary aspects of testing such as the objectives behind testing, the persons involved in testing, the instruments used and the testing procedures. B.3 presents an overview of glass box and black box testing techniques as they are practised by the SE community. B.4 presents a new model of test types which applies SE testing techniques to user-oriented testing and B.5 produces an overview of the instruments that are generally used for testing, their merits and drawbacks.

B.3 Survey of glass and black box testing techniques

Most of today's software testing techniques somehow go back to the early attempts to define the nature of software tests of the mid seventies. (Miller and Howden, 1981) is a comprehensive tutorial of software testing and validation techniques that takes up the most important approaches to testing and validation between 1974 and 1981. However, though the problem of testing has steadily gained importance, there is astonishingly little progress to be noted concerning the basic methodology and the techniques offered. Obviously, development in the area of testing techniques cannot keep pace with the increasing speed in the development of new hard- and software. This, however, is highly problematic because today's software systems are n-times more complex than their predecessors of the seventies. Despite the frequent usage of various tools which support the software development process to a large extent, the increasing complexity of systems leads to an increasing probability of software errors.

Glass box testing requires the intimate knowledge of program internals, while black box testing is based solely on the knowledge of the system requirements. Glass box (or white box) testing involves the knowledge of the program internals (Howden, 1980, 162), (Thaller, 1993, 135), (Bukowski, 1987, 370), and (Musa, Iannino and Okumoto, 1987, 521). Being primarily concerned with program internals, it is obvious in SE literature that the primary effort to develop a testing methodology has been devoted to glass box tests. However, since the importance of black box testing has gained general acknowledgement, also a certain number of useful black box testing techniques were developed.

The following survey will briefly outline the most important glass and black box testing techniques. Each of the different techniques as they are presented in the following has its own merits and drawbacks and are therefore mostly applied in combination rather than exclusively. Black box testing mainly requires the knowledge of the system requirements (Thaller, 1993, 135), (Musa et al., 1987, 521) and (Howden, 1980, 162).

B.3.1 Glass box testing

Glass box testing has traditionally been divided up into static and dynamic analysis (Hausen and Müllerburg, 1982, 119,122).

B.3.1.1 Static analysis techniques

The only generally acknowledged and therefore most important characteristic of static analysis techniques is that the testing as such does not necessitate the execution of the program (Hausen, 1984, 325). "Essential functions of static analysis are checking whether representations and descriptions of software are consistent, noncontradictory or unambiguous" (Hausen, 1984, 325). It aims at correct descriptions, specifications and representations of software systems and is therefore a precondition to any further testing exercise. Static analysis covers the lexical analysis of the program syntax and investigates and checks the structure and usage of the individual statements (Sneed, 1987, 10.3–3). There are principally three different possibilities of program testing (Sneed, 1987, 10.3–3), i.e.

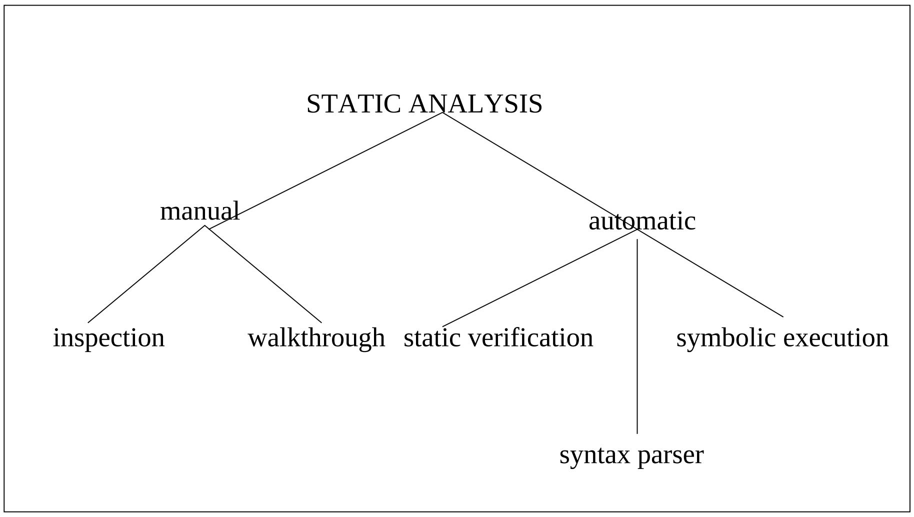


Figure B.1: Static Analysis Techniques

(i). program internally, to check completeness and consistency

(ii). considering pre-defined rules

(iii). comparing the program with its specification or documentation

While some software engineers consider it characteristic of static analysis techniques that they can be performed automatically, i.e. with the aid of specific tools such as parsers, data flow analysers, syntax analysers and the like (Hausen and Müllerburg, 1982, 126), (Miller, 1984, 260) and (Osterweil, 1984, 77), others also include manual techniques for testing that do not ask for an execution of the program (Sneed, 1987, 10.3-3). Figure B.1 is an attempt to structure the most important static testing techniques as they are presented in SE literature between 1975 and 1994.

Syntax parsers, which split the program/document text into individual statements, are the elementary automatic static analysis tools. When checking the program/document internally, the *consistency* of statements can be evaluated.

When performed with two texts on different semantic levels, i.e. a program against its specification, the *completeness* and *correctness* of the program can be evaluated. (Sneed, 1987, 10.3-6) and (Hausen, 1984, 328). This technique, which aims at detecting problems in the translation between specification and program realisation, is called static verification (Sneed, 1987, 10.3–3), and (Hausen, Müllerburg and Schmidt, 1987, 126). Verification requires formal specifications and formal definitions of the specification and programming languages used as well as a method of algorithmic proving that is adapted to these description means (Miller, 1984, 263) and (Hausen et al., 1987, 126). Static verification compares the actual values provided by the program with the target values as pre-defined in the specification document. It does not, however, provide any means to check whether the program actually solves the given problems, i.e. whether the specification as such is correct (Hausen et al., 1987, 126). The result of automatic static verification procedures is described in boolean terms, i.e. a statement is either true or false (Hausen et al., 1987, 127). The obvious advantage of static verification is that, being based on formal methods, it leads to objective and correct results. However, since it is both very difficult and time-consuming to elaborate the formal specifications which are needed for static verification, it is mostly only performed for software that needs to be highly reliable . Another technique which is normally subsumed under static analysis is called symbolic execution (Hausen, 1984, 327), (Miller, 1984, 263), (Hausen and Müllerburg, 1982, 117) and (Hausen et al., 1987, 127). It analyses, in symbolic terms, what a program does along a given path (Miller, 1984, 263). "By symbolic execution, we mean the process of computing the values of a program's variables as functions which represent the sequence of operations carried out as execution is traced along a specific path through the program." (Osterweil, 1984, 79). Symbolic execution is most appropriate for the analysis of mathematical algorithms. Making use of symbolic values only, whole classes of values can be represented by a single interpretation, which leads to a very high coverage of test cases (Hausen and Müllerburg, 1982, 117). The development of programs for symbolic execution is very expensive and therefore is mainly used for testing numerical programs, where the cost/benefit relation is acceptable.

The most important manual technique which allows testing the program without running it is software inspection (Thaller, 1994, 36), (Ackerman, Fowler and Ebenau, 1984, 14) and (Hausen et al., 1987, 126). The method of inspection originally goes back to Fagan (Fagan, 1976) who saw the practical necessity to implement procedures to improve software quality at several stages during the software life-cycle. In short a software inspection can be described as follows: "A software inspection is a group review process that is used to detect and correct defects in a software workproduct. It is a formal, technical activity that is performed by the workproduct author and a small peer group on a limited amount of material. It produces a formal, quantified report on the resources expended and the results achieved" (Ackerman et al., 1984, 14), (Thaller, 1994, 36), (Hausen et al., 1987, 126) and (Hausen, 1984, 324).

During inspection either the code or the design of a workproduct is compared to a set of pre-established inspection rules (Miller, 1984, 260) and (Thaller, 1994, 37). Inspection processes are mostly performed along checklists which cover typical aspects of software behaviour (Thaller, 1994, 37), (Hausen et al., 1987, 126). "Inspection of software means examining by reading, explaining, getting explanations and understanding of system descriptions, software specifications and programs" (Hausen, 1984, 324). Some software engineers report inspection as adequate for any kind of document, e.g. specifications, test plans etc. (Thaller, 1994, 37). While most testing techniques are intimately related to the system attribute whose value they are designed to measure, and thus offer no information about other attributes, a major advantage of inspection processes is that any kind of problem can be detected and thus results can be delivered with respect to every software quality factor (Thaller, 1994, 37) and (Hausen et al., 1987, 126).

Walkthroughs are similar peer review processes that involve the author of the program, the tester, a secretary and a moderator (Thaller, 1994, 43). The participants of a walkthrough create a small number of test cases by "simulating" the computer. Its objective is to question the logic and basic assumptions behind the source code, particularly of program interfaces in embedded systems (Thaller, 1994, 44).

B.3.1.2 Dynamic analysis techniques

While static analysis techniques do not necessitate the execution of the software, dynamic analysis is what is generally considered as "testing", i.e. it involves running the system. "The analysis of the behaviour of a software system before, during and after its execution in an artificial or real applicational environment characterises dynamic analysis" (Hausen, 1984, 326). Dynamic analysis techniques involve the running of the program formally under controlled circumstances and with specific results expected (Miller, 1984, 260). It shows whether a system is correct in the system states under examination or not (Hausen, 1984, 327).

Among the most important dynamic analysis techniques are path and branch testing. During dynamic analysis path testing involves the **execution** of the program during which as many as possible logical paths of a program are exercised (Miller, 1984, 260) and(Howden, 1980, 163). The major quality attribute measured by path testing is program *complexity* (Howden, 1980, 163) and (Sneed, 1987, 10.3–4). Branch testing requires that tests be constructed in a way that every branch in a program is traversed at least once (Howden, 1980, 163). Problems when running the branches lead to the probability of later program defects.

Today there are a number of dynamic analysers that are used during the software development process. The most important tools are presented in Table B.1 (Thaller, 1994, 177) :

Type of Dynamic Analyser	Functionality of Tool
test coverage analysis	tests to which extent the code can be checked by glass box techniques
tracing	follows all paths used during program execution and provides e.g. values for all variables etc.
tuning	measures resources used during program execution
simulator	simulates parts of systems, if e.g. the actual code or hardware are not available
assertion checking	tests whether certain conditions are given in complex logical constructs

Table B.1: Dynamic Analysis Tools

B.3.1.3 Generation of test data in glass box tests

The selection and generation of test data in glass box tests is an important discipline. The most basic approach to test data generation is **random testing**. For random testing a number of input values are generated automatically without being based on any structural or functional assumption (Sneed, 1987, 10.3-4) and (Bukowski, 1987, 370). There are also two more sophisticated approaches to test data generation, i.e. **structural testing** and **functional testing**. “Structural testing is an approach to testing in which the internal control structure of a program is used to guide the selection of test data. It is an attempt to take the internal functional properties of a program into account during test data generation and to avoid the limitations of black box functional testing” (Howden, 1980, 162). Functional testing as described by Howden (1980) takes into account both functional requirements of a system and important functional properties that are part of its design or implementation and which are **not** described in the requirements (Howden, 1980, 162). “In functional testing, a program is considered to be a function and is thought of in terms of input values and corresponding output values.” (Howden, 1980, 162). There are tools for test data generation on the market that can be used in combination with specific programming languages. Particularly for embedded systems, tools for test data generation are useful, since they can be used to simulate a larger system environment providing input data for every possible system interface (Thaller, 1994, 178). In other words, if a system is not fully implemented or not linked to all relevant data sources, not all system interfaces can be tested, because no input values are given for non-implemented functions. Data Generation tools provide input values for all available system interfaces as if a real module was linked to it.

B.3.2 Black box testing

Black box testing implies that the selection of test data as well as the interpretation of test results are performed on the basis of the functional properties of a piece of software. Black box testing should not be performed by the author of the program (Thaller, 1994, 92) who knows too much about the program internals. In new testing approaches, software systems are given a third external party for black box testing after having successfully finished the internal glass box testing exercises.

Though centered around the knowledge of user requirements, black box tests do not necessarily involve the participation of users. Among the most important black box tests that do not involve users are functionality testing, volume tests, stress tests, recovery testing, and benchmarks (Thaller, 1994, 138–149). Additionally, there are two types of black box test that involve users, i.e. field and laboratory tests (Karat, 1990, 352) and (Crellin, Horn and Preece, 1990, 330). In the following the most important aspects of these black box tests will be described briefly.

B.3.2.1 Black box testing - without user involvement

The so-called "functionality testing" is central to most testing exercises. Its primary objective is to assess whether the program does what it is supposed to do, i.e. what is specified in the requirements. There are different approaches to functionality testing. One is the testing of each program feature or function in sequence (Musa et al., 1987, 521). The other is to test module by module, i.e. each function where it is called first (Thaller, 1994, 138).

The objective of volume tests is to find the limitations of the software by processing a huge amount of data (Thaller, 1994, 139). A volume test can uncover problems that are related to the *efficiency* of a system, e.g. incorrect buffer sizes, a consumption of too much memory space, or only show that an error message would be needed telling the user that the system cannot process the given amount of data.

During a stress test, the system has to process a huge amount of data or perform many function calls within a short period of time. A typical example could be to perform the same function from all workstations connected in a LAN within a short period of time (e.g. sending e-mails, or, in the NLP area, to modify a term bank via different terminals simultaneously).

The aim of recovery testing is to make sure to which extent data can be recovered after a system breakdown. Does the system provide possibilities to recover all of the data or part of it? How much can be recovered and how? Is the recovered data still correct and consistent? Particularly for software that needs high *reliability* standards, recovery testing is very important.

The notion of benchmark tests involves the testing of program *efficiency*. The efficiency of a piece of software strongly depends on the hardware environment and therefore benchmark tests always consider the soft/hardware combination (Thaller, 1994, 147). Whereas for most software engineers benchmark tests are concerned with the quantitative measurement of specific operations (Thaller, 1994, 146), some also consider user tests that compare the efficiency of different software systems as benchmark tests (Lewis, Henry and Mack, 1990, 337–343). In the context of this document, however, benchmark tests only denote operations that are independent of personal variables.

B.3.2.2 Black box testing - with user involvement

For tests involving users, methodological considerations are rare in SE literature. Rather, one may find practical test reports that distinguish roughly between field and laboratory tests (Karat, 1990), (Crellin et al., 1990) and (Moll and Ulich, 1988). In the following only a rough description of field and laboratory tests will be given. For details see B.4.1.

In field tests users are observed while using the software system at their normal working place. Apart from general *usability*-related aspects, field tests are particularly useful for assessing the *interoperability* of the software system, i.e. how the technical integration of the system works. Moreover, field tests are the only real means to elucidate problems of the organisational integration of the software system into existing procedures. Particularly in the NLP environment this problem has frequently been underestimated. A typical example of the organisational problem of implementing a translation memory is the language service of a big automobile manufacturer, where the major implementation problem is not the technical environment, but the fact that many clients still submit their orders as print-out, that neither source texts nor target texts are properly organised and stored and, last but not least, individual translators are not too motivated to change their working habits.

Laboratory tests are mostly performed to assess the general *usability* of the system. Due to the high laboratory equipment costs laboratory tests are mostly only performed at big software houses such as IBM or Microsoft. Since laboratory tests provide testers with many technical possibilities, data collection and analysis are easier than for field tests.

To conclude, apart from the above described analytical methods of both glass and black box testing, there are further constructive means to guarantee high quality software end products. Among the most important constructive means are the usage of object-oriented programming tools, the integration of CASE tools, rapid prototyping, and last but not least the involvement of users in both software development and testing procedures (Thaller, 1993, 157).

The above survey of glass and black box testing methods has shown that although considerable work on development-oriented testing practices exists, there is much less work to date on user-oriented evaluation methods. We hope that this report will go some way to fill that lacuna.

B.4 A user-oriented model of test types

This section has two principal motivations, i.e. (i) to provide a comprehensive and theoretically sound definition of types of test which allow the judgement of the performance of a piece of software as seen from the eyes of the user and (ii) to give some practical advise to all those people who need to assess the quality of a certain piece of software for a particular reason.

Considering both methodological attempts to define software evaluation and practical test reports in the broad software engineering area, one may roughly distinguish between three principal motivations behind testing, i.e.

(i). to assess the appropriateness of a piece of software for every-day work

(ii). to examine the behaviour of software under specific conditions

(iii). to check the actual functionality of a piece of software

While both software developers and users may share the principal motivation behind testing to a certain extent, it is obvious that the actual testing procedures will differ largely. Despite this and the fact that there is no terminological consensus among leading software engineers (Sneed, 1987), (Musa et al., 1987), (Hausen, 1984), (Hausen et al., 1987), (Hausen and Müllerburg, 1982), (Hausen et al., 1987), (Fagan, 1976) and (Howden, 1980), it is necessary that the extensive testing experience of software engineers be considered when developing a model for user-oriented testing, while at the same time introducing new user-oriented testing terminology. The terminology used here partly goes back to (Ahmad, Holmes-Higgin, Rogers, Höge, Le-Hong, Huwig, Kese and Mayer, 1993, 9). Accordingly, the three major test types discussed here will be called scenario tests B.4.1, systematic tests B.4.2, and feature inspection B.4.3.

Apart from the principal objective of evaluation, the practical choice of test type and instruments to be used depends on various factors such as (i) the testing budget, (ii) the testing environment, i.e. whether a professional testing team or a user company undertakes the software tests; (iii) the availability of evaluation and testing expertise and equipment; (iv) time that is going to be invested in testing; and (v) the availability of users as subjects for tests (Crellin et al., 1990, 331). In addition to a theoretical definition of the different types of test, the practical guidelines provided will discuss the typical ISO quality characteristics which can be assessed, the (type of) personnel involved, the (type of) instruments employed, the (type of) results achieved and their constraints on objectivity and informativeness.

The following model of test types is meant as a starting point for anyone who has to perform user-oriented software tests. For each practical evaluation scenario with its specific environmental constraints, however, individual decisions have to be made concerning the most appropriate test type(s). For most cases it will become obvious that "in order to evaluate the usability of a system it is often necessary to use several methods in combination." (Vainio-Larsson, 1990, 324) and (Hausen et al., 1987, 141).

B.4.1 Scenario tests

Though "the need to test systems in real work environments is receiving increased attention" (Karat, 1990, 352), there has been hardly any methodological attempt to define the exact nature of these kinds of tests, which have so far been mainly the domain of software developers.

The term "scenario" has entered software evaluation in the early 1990s (Lewis et al., 1990, 337). A scenario test is a test case which aims at a realistic user background for the evaluation of software as it was defined and performed, for instance in the TWB projects (Höge, Wiedenmann and Kroupa, 1991, 12), (Höge and Kroupa, 1991, 10), (Ahmad et al., 1993, 322), (Höge, Hohmann and Le-Hong, 1993, 166) and (Le-Hong, Höge and Hohmann, 1992, 29). and later also adopted by the EAGLES evaluation group. It is an instance of black box testing where the major objective is to assess the suitability of a software product for every-day routines. In short it involves putting the system into its intended use by its envisaged type of user, performing a standardised task.

One may roughly distinguish between five major phases of scenario testing, i.e. test planning, test preparation, testing, data analysis and reporting. During each of these phases a number of problems have to be tackled by the testing team before the next phase can start. Figures B.2, B.3, B.4, B.5, and B.6, briefly outline the major problems of each phase and the corresponding tasks.

problem	**task**
costs	define evaluation project budget
software	decide whether system test or module test
type of scenario test	decide whether field or laboratory
location	decide where define technical environment
users	decide how many users, who and why
time	select week/day(s)
evaluators	decide how many evaluators and who

Table B.2: Central Problems of the Test Planning Phase

It is of utmost importance to note that tests involving users pose the most difficult problems in the assessment phase, because the high number of personal variables such as education, experience, age, motivation, day-time, work-load etc. is likely to blur the objectivity of test results, while at the same time they rank high concerning aspects such as informativeness.

B.4.1.1 Types of scenario tests

Two different possibilities to perform scenario tests are reported in software engineering literature, i.e. field and laboratory tests (Karat, 1990, 352), (Crellin et al., 1990, 330) and (Oppermann, Murcher, Pateau, Pieper, Simm and Stellmacher, 1988, 13) which involve different testing environments, tasks, requirements on test system, user participation, instruments, testing expertise, and, last but not least, time and money constraints.

B.4.1.1.1 Field tests A field test is a type of scenario test in which the **testing environment** is the normal working place of the user who is observed by one or more evaluators taking notes, time etc. From a psychological point of view, the field test is considered to be the least obtrusive test in that it involves basically the same physical and social environment factors as normal work does (Karat, 1990, 352) and (Oppermann et al., 1988, 12). Among the physical environment factors which are likely to influence the behaviour of the user are the layout of the office space, crowding and

problem	task
quality requirements	study user requirements select relevant quality characteristics define metrics and target values
test task	define task and sub-tasks
test data	decide whether test corpus or collection or suite elaborate test data
instruments	decide which instrument: - questionnaires - interviews - test programs - observation - think-aloud - checklists elaborate instruments integrating metrics
equipment	get equipment prepare equipment
test procedure	define tasks of evaluators define steps of testing exercise
duration	define start/end of each step elaborate time schedule
test-plan	elaborate test-plan fixing above-mentioned aspects

Table B.3: Central Problems of the Test Preparation Phase

noise level. The most important social environment factors are office atmosphere and the normal pace of work (people stopping by and requesting information etc.) (Karat, 1990, 352). However, despite the advantage of displaying the every-day physical and social environment factors, a certain variance in behaviour can result from the psychological effects of being observed while working.

The **task** to be performed by different users during the field test should be standardised which guarantees that every user will encounter the same kind of problems and will have to perform similar operations to succeed (Moll and Ulich, 1988, 73). Ideally the overall test task fits well into the organisational routine of the user's every day work and was developed beforehand in consultation with a number of users of the same environment (Karat, 1990, 352). An obvious advantage of field tests as compared to laboratory tests is that the test task can include problems of data transfer between the test-system and existing systems. To ease evaluation, the overall test task needs to be divided into sub-tasks, each identifying an operational unit of performance. For each sub-task the metrics that are of interest should be defined beforehand, so that the evaluator's attention is automatically focused on particular aspects of performance.

Closely related to the problem of the test task is the requirements on the **system** under testing. It is obvious that, if the test task can be considered as part of the daily organisational routine, the software system under testing needs to be in a highly operable condition. Thus field tests are most beneficial if the systems under testing are β-versions of products to be launched in the near future, or off-the-shelf products. The more the system presupposes a deviation of the task from the normal routine, the less informative are the results of the field tests.

For both kinds of scenario tests it is important that a representative number of **users** participate in the tests. There are a great number of personal variables involved that can have a decisive influence on the performance of the system, i.e. in all cases computer literacy, motivation or day-time (Oppermann et al., 1988, 12), and for the more complex NLP applications education, experience and

problem	**task**
time-management	make sure that time schedule is kept
organisation	distribute tasks among users and evaluators
observation	note aspects of user behaviour
trouble shooting	react on problems document deviations from test plan

Table B.4: Central Problems of the Testing Phase

problem	**task**
data viewing	analyse data and apply relevant metrics
data collection	consider results of relevant metrics of all subjects
calculation	define value types for all metrics calculate statistical averages calculate statistical variance

Table B.5: Central Problems of the Data Analysis Phase

expertise are of great importance. The naturally subjective results arrived at by means of scenario tests need to be statistically interpreted and "objectivized". The organisational environment of field tests, which do not involve much extra expenditure for equipment etc., normally allows the participation of more users in tests than in laboratory tests of comparable costs.

The **instruments** commonly used in field tests range from the simple observation of users and noting their behaviour and interaction times on evaluation checklists (Karat, 1990), (Höge et al., 1991, 17), (Höge, Hohmann and Mayer, 1992, 7–14), and (Ahmad et al., 1993, 14) to pre-and post-testing interviews (Moll and Ulich, 1988, 72), (Höge et al., 1991, 17), (Höge et al., 1992, 7–14) and (Ahmad et al., 1993, 14), think aloud (Moll and Ulich, 1988, 74) and (Vainio-Larsson, 1990, 325), and, last but not least, logfile recording, a facility for recording all keystrokes during the user interaction in a separate file (Vainio-Larsson, 1990, 325). The choice of instruments depends on various factors such as time and money constraints, technical facilities, evaluation expertise etc. Due to the limited possibilities of retrospective data analysis present in field tests, it is of utmost importance that the data gained with the aid of the different instruments (notes on user behaviour, interaction etc.) be analysed right after finishing the test, because otherwise important contextual information is likely to get lost. The **evaluation** setup of field tests generally puts heavy demands on the expertise and experience of the evaluator. Test planning is rather difficult for field tests, since the natural environment involves a great number of variables which cannot easily be determined beforehand. The optimal point in time needs to be found for integrating a new piece of software into the daily routines. The test preparation phase for field tests is more time consuming than for its laboratory counterpart, since user behaviour needs to be anticipated to a certain extent in order to pre-define applicable metrics. Similarly, the testing phase is highly demanding in field tests, since it is very difficult to filter out important from non-important user behaviour during the test. Also, from the organisational point of view, a field test is only satisfactory, if the normal working routine is interrupted as little as possible during the test. Experience together with well prepared evaluation instruments such as checklists etc. are among the most decisive means to guarantee that the results of field tests are satisfactory. Whereas laboratory tests provide the evaluator with various possibilities to record and re-play the different test situations, evaluators in field tests mostly have to rely on what they identify as important information during the various situations in a test. This makes the data analysis phase of field tests particularly difficult, since if not performed directly

problem	task
documentation	document test-bed precisely document all decisions taken during all phases document all deviations from test plan provide total of testing data as appendix
evaluation	justify all interpretations of results summarise results

Table B.6: Central Problems of the Reporting Phase

after the completion of the test, time is likely to blur the situational context in which results were achieved and thus the probability of inadequate data interpretation is high.

The final costs of a scenario test can be calculated from two major types of cost, i.e. personnel and equipment. On the personnel side, expenses are calculated by means of the PH (person hour) rates for the evaluators and users involved in the overall test. As demonstrated above, field tests involve a great number of PH for evaluators, particularly in the test planning and preparation phases. Mainly due to the natural environment, the number of users participating in tests is normally higher than in laboratory tests. However, the time that has to be invested for each participant, including introduction, test and post-testing interview, is much less than in laboratory tests. The major difference in costs between field and laboratory tests, however, lies in the equipment. From the equipment side, it is obvious that field tests ask for comparatively little expenses, because almost no investment in additional technical evaluation instruments is obligatory. Thus, given the availability of a certain expertise of evaluators, it is obvious that field tests mostly invoke less costs than their laboratory counterpart (Karat, 1990, 355).

B.4.1.1.2 Laboratory tests A laboratory test is an instance of a scenario test in which the **testing environment** includes a number of isolated users who perform a given task in a test laboratory which offers a great variety of data collection techniques (Karat, 1990, 352). When interpreting test results, it has to be kept in mind that "the end user's social and physical environment is not replicated in the laboratory, and these factors influence the way an end user works with an application." (Karat, 1990, 352) and (Oppermann et al., 1988, 13). On the other hand, laboratory tests allow the participation of system developers more easily than field tests, which gives a certain impetus to remedy problems in development projects (Karat, 1990, 352). Like field tests, laboratory tests should cover standardised test tasks which allow the comparison of user behaviour. On the one hand, the isolated testing environment in a laboratory does not allow any assessment of data transfer routines and therefore has a rather modular character. For instance, in the case of performing tests with a translator's workstation, a laboratory test cannot test whether the terminology provided by the existing mainframe terminology management system at the customer's site can be integrated into the editing environment. On the other hand, since the task does not have to fit into the routines of every-day work, it is possible in laboratory tests to define less comprehensive test tasks as frequently needed for testing system prototypes. The nature of the test task, i.e. whether the task is representative or simply has testing character, also influences the user's behaviour during the test (Oppermann et al., 1988, 13). Since there is a much greater variability in the definition of the test task, laboratory tests are particularly useful if the **system** under testing is not fully operable. Most of the bigger software houses, therefore, perform a number of laboratory tests at different stages of the software life-cycle rather than putting much effort into the design of comprehensive field tests.

In laboratory tests **users** need to be highly motivated in order to deliver useful test results. This is due to the fact that subjects have to invest a certain amount of time to get used to the new working environment and the demands the tests put on them. Moreover, due to the extra time needed for travel as well as for the introduction of subjects, normally less users participate in laboratory tests than in field tests.

The artificial environment in laboratory tests allow the usage of a great number of technical **instruments**. Well-equipped laboratories offer one-way mirrors, video and audio recording facilities as well as different logging programs. For details on logging programs see B.5.1.2.

Cf. also (Karat, 1990, 351) and (Crellin et al., 1990, 332). In combination with these technical instruments, checklists and questionnaires have proved to be useful in order to cover the maximum amount of relevant information (Vainio-Larsson, 1990, 325) and (Karat, 1990, 351). In laboratory tests the planning phase is comparatively easy, since the test does not have to fit into any working routine. Metrics that are defined prior to the test in the test preparation phase can be enriched with additional aspects of performance when **retrospectively** analysing the data. The testing phase is also less complicated than in field tests, because time-management, organisation and trouble shooting are not crucial for the successful completion of the laboratory test which lays no claim to resemble normal work. Making use of various technical devices for data storage and retrieval, the data analysis phase becomes less difficult but at the same time often much more time-intensive than in its counterpart.

The **costs** that need to be calculated for laboratory tests are reported to be around 4 times higher than those of comparable field tests (Karat, 1990, 352). The major factor in this calculation goes back to the very expensive maintenance of a laboratory with its various technical devices. Though evaluators in field tests invest more time in test planning and preparation, the person hours invested by evaluators in laboratory tests are comparable, since the data analysis phase is much more time-intensive than in field tests. This is mainly due to the fact that the more technical instruments are involved in data recording, the more data is available. The correlation of results from different data storage devices is both difficult and time-intensive. Moreover, most of the data is not put down during the actual testing as in field tests, but rather during a great number of retrospective analysis procedures. On the user side, normally less PHs are invested for actual testing than for organisational tasks such as the selection of subjects, travel and introduction into the new environment.

Figure B.7 summarises the major differences between the two types of scenario test as discussed above.

	FIELD TEST	**LABORATORY TEST**
testing environment	normal working place - least (but still slightly) obtrusive - same physical/social environment factors	laboratory - controlled - new working environment - integration of developers into tests possible
test task	representative integrated task - fits into every-day routine - includes problems of data transfer	individual tasks - possible to test specific modules only
test system required	operable system or beta version	prototypes or operable systems
users	more users/budget	less users/budget
instruments	direct observation think-aloud checklisting pre- and/or post-testing interviews logging programs	indirect observation - one-way mirrors - video recording - audio recording think-aloud logging programs
evaluation test planning phase:	complex	straightforward
test preparation	time-intensive difficult (training observers)	less time-intensive straightforward
testing	very difficult	
data analysis	brief	exhaustive
costs approx. % of total PH test planning phase:	10%	5%
test preparation	50%	30%
testing	10%	10%
data analysis	20%	45%
reporting	10%	10%
costs equipment	negligible	high
costs of test types compared	25%	100%

Table B.7: Field Test - Laboratory Test - A Comparison

Concluding the discussion of types of scenario test, it is important to note that experience shows that it is often very useful to combine the two types of scenario test, as Karat reports: "By combining the use of field and laboratory tests and testing prototype and integrated code, usability staff were able to collect complementary information that together provided a more complete understanding of the end user, work contest, and panel issues and resulted in a better final interface design than would have been possible without iterative testing using different methodologies" (Karat, 1990, 352).

B.4.1.2 Types of metrics and results in scenario tests

Of all test types it is the scenario test that can provide most detailed information on the quality subcharacteristics *understandability*, *learnability* and *operability* subsumed under *usability*. (ISO, 1991). Additionally, scenario tests can provide information on *suitability*, *accuracy*, *interoperability* (subcharacteristics of *functionality*), *time behaviour*, *resource behaviour* (subcharacteristics of *efficiency*), *changeability* (subcharacteristic of *maintainability*) and *adaptability* (subcharacteristic of *portability*).

When elaborating metrics for scenario tests a top-down approach is most appropriate. The evaluator then starts with a top-level item of the requirements specification and considers each of the different quality characteristics and sub-characteristics as they are provided by ISO 9126. For each characteristic which is identified as important for the particular application, the evaluator needs to describe how the system should ideally perform concerning system and data dimension. Having fixed the expected performance of the system, metrics have to be found which allow the measurement of system performance. Note that involving subjective users, most metrics in scenario tests are only indirectly quantifiable, i.e. only by involving a number of users, the results, which are in the first instance subjective qualitative data, can be statistically objectivised and quantified. Metrics that are typically reported to be applied in scenario tests are *time on task* (Karat, 1990, 353), (Lewis et al., 1990, 338) and (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10), *completion rate*, *error free rate*, *time needed for training programme* (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10), *frequency of help/documentation use*, etc. (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10).

The results of scenario tests mainly need to be considered in the light of *objectivity*. As pointed out before, the results of neither field nor laboratory tests can be considered objective, since both go back to the participation of subjective individuals. A single person cannot be considered objective. The most commonly used techniques to reduce subjectivity in scenario tests, therefore, are to calculate averages and variances on a sufficiently large number of subjective judgements, while trying to avoid inferences with other systems. However, striving for a "cleanroom" approach for scenario tests by selecting test persons and subjects that do not have inferences with other systems, is dangerous, since while achieving more objectivity, the results are likely to become less representative.

B.4.2 Systematic testing

Under the term systematic testing all testing activities will be subsumed that examine the behaviour of software under specific conditions with particular results expected. The term "systematic testing" was shaped by (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10). Whereas the objectives behind scenario testing ask for the integration of users into the testing exercise, systematic tests can be performed solely by software engineers and/or user representatives. B.3 describes a number of objectives behind testing that need further elaboration in order to satisfy the needs of user-oriented testing. There are three objectives that are particularly relevant for user-oriented testing, i.e.

(i). examining whether the software fulfils pre-defined tasks;

(ii). examining whether the functions offered work properly; and

(iii). examining the performance of the system.

Accordingly, user-oriented systematic testing will be split up into task-oriented testing (B.4.2.1), menu-oriented testing (B.4.2.2), and benchmark testing (B.4.2.3).

B.4.2.1 Task-oriented testing

Task-oriented testing is performed to examine whether a piece of software actually fulfils pre-defined tasks. These tasks may either be stated in the requirements specification document as in software development projects, or may be implied by third parties as for instance in consumer reports. Task-oriented testing is related to scenario testing in that its major purpose is to assess the overall functionality of the system by means of relevant data inputs, as well as to examine the quality of the data output. There are several reasons why one may decide for task-oriented testing instead of performing scenario tests, i.e.

(i). restricted budget

(ii). time constraints

(iii). no users available

(iv). no laboratory available

(v). functionality of prototypes restrictive in performance (Crellin et al., 1990, 331)

(vi). only individual tasks can be performed by the prototype

As for scenario testing, for task-oriented testing a top-down approach is most adequate. The final success of task-oriented testing lies in the exhaustive definition of a representative number of test tasks and subtasks as they are likely to occur in every-day routines and are covered by the system. For this purpose, the evaluator needs to consult a number of users and discuss the technical and organisational constraints of their every day work as well as the type of task they have to perform by the aid of the software. Having performed task-oriented testing, it is important that the data output be examined thoroughly and discussed with the end users of the system.

Task-oriented testing can be carried out during the software development process at any stage of the software life-cycle as well as with any off-the-shelf software product. Generally, task-oriented testing does not ask for extensive test planning. Since task-oriented testing does not involve users as subjects (users should still be involved in the definition of the test task), the overall organisation of the test is less demanding. The **testing environment** is normally the working place of the evaluator and is principally not relevant for the interpretation of results.

The major effort of the test preparation phase lies in the definition of the test tasks. Whereas in scenario tests, the test task needs to be standardised, task-oriented testing can define a broad range of test tasks which may all be relevant to one user or the other. While in scenario testing the sub-tasks have to be defined beforehand, task-oriented testing leaves more space concerning an investigation of the possible ways of performing a given task. Whereas in field tests, for instance, problems in the performance of the test task lead to an interruption and therefore a rather costly failure of the whole testing exercise, task-oriented testing allows a repetition of test tasks, while documenting the problems encountered. In addition to the metrics which could be defined in the test preparation phase, it is often the case in task-oriented testing that additional metrics can be applied or additional data be tested, once the detailed functionality of the system is accessible. The number of testing instruments that are applied during task-oriented testing is restricted to checklists containing the tasks, sub-tasks and related metrics and reporting instruments such as result reports (cf. result sheets in (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10), or metric-level tally sheets in (Murine and Carpenter, 1983, 374)), and test problem reports (Deutsch, 1982, 290) (cf. failure sheets in (Höge et al., 1992, B-3/1–B-3/7)).

Having prepared the test tasks, data and metrics in the preparation phase, the actual testing phase is not very demanding and can be performed by anyone who has some knowledge of **evaluation** and the definition of metrics, i.e. by software developers, users or management.

The **costs** of task-oriented testing are comparatively small and the PH invested mainly depend on the number of tasks tested. Apart from the technical environment of the evaluator (hard- and software) no extra investment into testing equipment or instrument is necessary for task-oriented testing.

B.4.2.1.1 Types of metrics and results in task-oriented testing The primary quality characteristic under investigation in task-oriented testing is *functionality*. In this sense task-oriented testing comes close to what is called "functionality testing" in B.3.2.1, i.e. investigating whether the program does what it is supposed to do. Being able to test a great number of different tasks, the *suitability* of the software, i.e. the presence and appropriateness of a set of functions for specified tasks, can be closely examined (ISO, 1991, A.2.1.1). Another important aspect of task-oriented testing is the examination of the quality of data output as it is described under *accuracy*, the "attributes of software that bear on the provision of right or agreed results or effects." (ISO, 1991, A.2.1.2). When performed at the final installation place of the software, task-oriented testing can also deliver valuable results concerning the *interoperability* of the software. For this purpose, the tasks must cover the communication with other applications and/or users within the given environment. In addition the *usability* of the system can be assessed in a different way than in scenario testing: one of the most frequently applied metrics of *operability* in task-oriented testing, for instance, is the counting of steps necessary to perform a certain task (how many actions are required to perform a given subtask).

The metrics applied in task-oriented testing deliver mostly boolean (presence or absence of functions), quantitative (e.g. number of steps etc.) or classificatory (how well a function performs a task) values. While both boolean and number are objective values, classificatory values are based on the subjective impression of the evaluator. While in scenario tests the originally subjective statements of users can be "objectivised" by involving a representative number of users in the tests, task-oriented testing is normally performed by only one (better two) evaluators and thus has to be seen in the light of subjectivity. While the boolean values that determine the presence or absence of functions can be considered highly reliable, the classificatory values that denote the quality of the implementation of the individual functions are not necessarily repeatable when involving a different evaluator and can therefore not be considered particularly reliable.

B.4.2.2 Menu-oriented testing

While both scenario and task-oriented testing are mainly geared to examine the handling and functionality of the software, the philosophy behind menu-oriented testing comes closest to the developer's principal aim, i.e. the discovery of software problems. The basic idea behind menu-oriented testing, i.e. the "testing of each program feature or function in sequence" (Musa et al., 1987, 521), is prominent in many glass and black box testing techniques. As in the structured walkthrough (cf. B.3.1.1), the software is examined from top to toe, considering each individual function as it is sequentially offered in the menu bar. Instead of walking through the code, during menu-oriented testing, the evaluator "walks through" the executables, following each possible path of program execution. Cf. path testing in section B.3.1.2 B.3.1.2. Thus, while in both scenario and task-oriented testing only particular functions are performed, namely those that are necessary to perform the test tasks, in menu-oriented testing each function of the software is executed at least once.

Similar to task-oriented testing, menu-oriented testing can be performed at any stage of the software life-cycle as well as with off-the-shelf products. For menu-oriented testing little time has to be invested in the test planning and test preparation phases. No users have to be found and the test does not have to fit into any operational **environment**. It can easily be performed in isolation on the evaluator's computer, which, however, reduces the number of metrics that can be applied (neglecting for instance aspects of interoperability).

Compared to the extensive test preparation phase of both scenario and task-oriented testing, the activities of the test preparation phase in menu-oriented testing are reduced to a minimum. No specific test **task** needs to be defined beforehand and, therefore, the time-consuming consultation of users is not necessary. Instead of following pre-defined tasks, the evaluator investigates any possible way of handling the system. Since the test is not performed along a certain pre-defined task, only few principal metrics can be defined in advance. The preparation of the **instruments** necessary for menu-oriented testing - mainly result reports and software trouble reports - is neither difficult nor time consuming.

The major effort in menu-oriented testing clearly has to be spent during the testing phase, since apart from applying pre-defined metrics, it may be necessary that the evaluator has to develop metrics and elaborate test data, perform tests and document the results on an ad hoc basis while executing the software. Only when executing a certain function, the evaluator can guess which data is needed to perform certain operations with the functions offered in the menu bar (the WHAT HAPPENS IF ... test). In the case of a terminology elicitation system, for instance, a volume test can be performed by executing the option “do concordance” with an unusually big file. Cf. (Hohmann, Le-Hong and van der Horst, 1994, 71), where the result of this type of volume test with the terminology elicitation system “System Quirk” is documented. For a translation memory a stress test could be performed, accessing, for instance, a certain number of parallel texts by different users on the LAN at the same time. Recovery tests could be performed with a termbank when e.g. simulating a system breakdown (e.g. on PCs by means of pressing control/alt/del keys) before and after having properly saved terminology modifications. It is obvious that, compared to task-oriented testing, successful menu-oriented testing presupposes a good deal of both **evaluation** expertise and experience.

The **costs** of menu-oriented testing mainly lie in the recruitment of excellent evaluation personnel that is capable of the ad hoc generation of metrics and data. Similar to task-oriented testing no investment is necessary for additional testing instruments.

B.4.2.2.1 Types of metrics and results in menu-oriented testing In menu-oriented testing nearly all user-related quality characteristics can be assessed. While *suitability*, *understandability*, *operability* and *learnability* should also be assessed by means of scenario or task-oriented testing, the major benefit of menu-oriented testing lies in delivering results on the system's *reliability*, including characteristics such as *maturity*, *fault tolerance* or *recoverability*. On the *functionality* side, menu-oriented testing delivers valuable results concerning the system's *compliance* with other standards, e.g. whether a windows application consistently makes use of the same windows system messages as other applications do, or whether the system is internally consistent with its labelling of functions and processes etc. Also system *security* is one of the characteristics that only menu-oriented testing can sufficiently investigate, since unlike in task-oriented and scenario testing, it is certain that every function is performed and checked.

The results achieved by means of menu-oriented testing are mostly qualitative and descriptive in the sense that any observation related to the different quality characteristics is directly noted on the result report. If pre-defined metrics are applied or metrics performed such as volume, stress or recovery tests, the results are either boolean, classificatory, or, in rare cases also number. For qualitative statements, the objectivity is comparatively low, since menu-oriented testing is performed by individuals. Boolean and quantitative results delivered by means of volume, stress or recovery tests can be considered objective, since other evaluators are likely to achieve the same results when keeping data input and environment of the system constant.

B.4.2.3 Benchmark testing

Benchmark tests examine the performance of systems. The notion of performance can be applied either to individual functions, modules or to the overall system. In the strict technical sense, a benchmark test is the measurement of system performance without being dependent on personal variables (Thaller, 1994, 146). Thus, following the narrow definition of benchmark, there are very few possibilities of applying benchmarks on the module or even system level of interactive systems. Different views on the definition of benchmark can be found in (Lewis et al., 1990, 337) and (Oppermann et al., 1988, 12). Examples of benchmark tests in the NLP area are rather on the function level, e.g. the measurement of success rates for automatic terminology retrieval functions, the measurement of translation retrieval rates for translation memories, the measurement of time for the parsing of a text etc.. Benchmark tests allow the comparison of the performance of different tools. When performing the same benchmark with different systems, it has to be kept in mind that both system parameters and environment variables are kept constant. The comparison of the benchmark results only makes sense if different translation memories, for instance, have access to the very same background material and are tested with the very same test text.

For benchmark tests, the test preparation phase is most decisive, since it involves the identification of appropriate units or functions which operate without being influenced by the evaluator, the selection of test data, and the fixation of the measurement technique. The testing **environment** is a decisive factor influencing the result of the benchmark and thus needs to be documented carefully. (Cf. B.5.2.1). Typical instruments applied for benchmark tests are checklists that cover the quality characteristic, the benchmark, measurement technique and results, or, if a benchmark involves the execution of more than one function, it is useful for testing purposes if shellscripts are programmed that include the sequence of function calls as they are performed for the benchmark test.

B.4.2.3.1 Types of metrics and results in benchmark tests The major quality characteristic assessed by means of benchmark tests is *efficiency*. Apart from *time* and *resource behaviour* (ISO, 1991, A.2.4.1–A.2.4.2), an important quality characteristic that falls under efficiency is *output behaviour*. Particularly in the NLP area a system's capability to produce a certain amount of output in a given time is often measured in benchmark tests.

The types of result achieved by means of benchmark tests are mostly numbers, e.g. the time needed to perform a certain function, the resources needed when performing a function, and, last but not least, the amount of output data produced in a given time.

B.4.3 Feature inspection

The aim of feature inspection is to be able to describe the technical features of a piece of software as detailed as possible, so that it allows the comparison between systems of the same type. Feature checklists are compiled in the awareness of the possibilities individual systems offer and with the aim of demonstrating the differences between similar tools. The results of feature inspection are meant to help consumers to decide which of the systems on the market are most appropriate for their particular environment.

The notion of "inspection" has been applied to this type of test because in the wider sense all manual testing techniques can be considered inspections (Thaller, 1994, 36), (Ackerman et al., 1984, 14) and (Hausen et al., 1987, 126). Moreover, feature inspection involves the comparison between a piece of software and a pre-defined feature checklist. (Cf. the notion of inspection in B.3.1.1 and "checklist-oriented testing" in (Oppermann et al., 1988, 13).) Also, similar to the original glass box inspection method, feature inspection does not necessarily involve the execution of the program. However, while its glass box counterpart is a means to actively seek for software problems, feature inspection has a more descriptive character, checking the availability of features rather than the absence of errors.

Feature inspection is a test type which asks for tremendous efforts mainly in the test preparation phase. Successful feature inspection depends mainly on the quality of the feature checklist along which the evaluator examines the software. Feature checklists incorporate the knowledge of what is important from the user's point of view with what is possible from the technical point of view. In order to be able to compile comprehensive feature checklists for a particular type of tool e.g. translation memories, the evaluator needs to study the user requirements including their organisational constraints and has to get acquainted with a broad range of systems in order to be able to grasp their underlying philosophy. The most appropriate approach to developing feature checklists is top-down, i.e. starting with principal functions before discussing their details. Any feature checklist in the context of evaluation needs to be both standardised in the sense that it should be independent of situational variables and open in the sense that it can cover different approaches to a problem without being prescriptive in nature. Since most feature checklists are based on the state of the art of development, they only describe features that are common technology. If, however, new technical solutions have been found to an old problem, these solutions are not likely to be instantly part of feature checklists. An example of a prescriptive feature checklist of translation memories, for instance, would ask whether and how many databases the system offers for storing and retrieving parallel text. If, however, a system does not store parallel text in databases but rather accesses the text in the normal working directories, the feature checklist would exclude this second type of translation memory which might be even more innovative than those using databases. The outcome of the feature inspection in this case could lead to a negative judgement of the tool, because of the absence of a feature which is normally rated high.

In the testing phase, the evaluator has to follow the top-down approach of the checklist and simply look through the software whether the features on the checklists are available or not. Thus the actual testing phase is less problematic, since the evaluator does not have to perform any comprehensive tasks or develop any metrics while testing as it is the case in systematic testing.

Both data analysis and reporting phases in feature inspection are reduced to a minimum, since the reporting is performed while testing and the results mostly do not allow any interpretation.

B.4.3.1 Types of metrics and results in feature inspection

Similar to glass box inspection, feature inspection can tackle every quality characteristic. (Cf. B.3.1.1.) The major focus, however, is on investigating the system's *functionality*.

Feature checklists are mostly designed to elicit boolean values, i.e. demonstrate the presence or absence of features. For tests delivering boolean values the *objectivity* can generally be considered high.

B.5 Survey of instruments

In this context the term *instrument* denotes any form of experiemental data collection that can be used in combination with the different test types as described above. The principal question is how to collect data that gives some hint on software performance. Most testing frameworks do not distinguish between test type and instrument which may lead to the incorrect conclusion that test type and instrument are inseparable units. Testing experience, however, shows that there are various possibilities to perform a particular type of test, making use of different test instruments (Crellin et al., 1990, 330). The choice of instruments depends, as previously stated, on various factors such as time and money constraints, testing experience, testing environment etc. Also, it has to be noted that many individual testing instruments are "impoverished" forms of data collection. However, when used in conjunction the sum is far greater than its parts (Crellin et al., 1990, 333).

There are basically two types of instrument, i.e. those used for data collection in the testing phase and those used for data reporting in the reporting phase. The present approach guides the user of the evaluation framework first through the selection of the appropriate test type (cf. B.4), over the choice of the suitable testing instruments (B.5.1), to the corresponding reporting instruments (B.5.2) needed for the particular evaluation purpose.

B.5.1 Testing instruments

In the testing phase, there are two major kinds of testing instruments, i.e. those that ask for a *manual* collection of data and those that perform *automatic* data collection.

B.5.1.1 Manual test instruments for user-oriented testing

The following survey will concentrate on describing the most important characteristics of the most prominent manual testing instruments - questionnaires, checklists, interviews, observations, think-aloud protocols - as they could be collected from a great number of practical test reports.

B.5.1.1.1 Questionnaires Questionnaires are frequently used for all phases of software development and evaluation (Oppermann et al., 1988, 10). Questionnaires are used to elicit both quantitative and qualitative data. For qualitative data often numerical rating scores are used that lead to an on the first sight "objective" numerical result (Crellin et al., 1990, 333) and (Oppermann et al., 1988, 10). The reliability of results arrived at by means of rating scales used in questionnaires strongly depends on the adequacy of the rating scale, and on both the number and representativity of persons questioned (Falkedahl, 1991, 25). Another often cited problem of questionnaires is that they are likely to deliver only those results that are welcome to the designers of the questionnaire, i.e. the choice of questions biases the results. This is due to the fact that the way of posing questions may implicitly suggest the "correct" answer (Oppermann et al., 1988, 10). Thus both validity and informativeness of questionnaire results are a matter of the adequacy of the questions and rating scales (King and Falkedal, 1990, 25).

Whereas paper questionnaires are still prominent when performing large scale surveys, for testing purposes the interactive on-line questionnaire is steadily gaining importance (Rushinek and Rushinek, 1985, 250) and (Crellin et al., 1990, 333). Interactive on-line questionnaires are linked to the software and at critical points in the performance of the program, a question pops up which the subject has to answer directly. Being linked to the program, it is impossible to use the software without answering the questions posed. The responses are then stored automatically and statistical calculations performed over the whole range of responses of different subjects (Rushinek and Rushinek, 1985, 251). On-line interactive questionnaires are sophisticated means to elicit information particularly on user-related quality characteristics such as *usability*, or *functionality*. This is due to the fact that the questions are posed directly when the user has performed the critical function, i.e. at the very moment when the impression of the software on the user is at its peak. At the same time interaction questionnaires disrupt the normal work flow of the user and thus the findings are related to the *usability* and *functionality* of individual modules rather than to the overall system.

The elaboration of on-line interactive questionnaires from scratch is time-consuming and presupposes a certain knowledge of object oriented programming. There are different shells that ease the elaboration of questionnaires mostly including a system usage log, interactive on-line questionnaire (plus rating scales) and data analysis tools. The setting up of an on-line interactive questionnaire by the aid of these shells is reported to need little expertise, time and money. Combining the use of different test instruments, a great amount of data can be collected. Having correctly set up the system, subjects can perform the test on their own, without the need of an evaluator being always present, which reduces the personnel costs and eases the test planning phase to a large extent, since no dates for tests have to be organised with a number of users who are able to perform the tests whenever they like. However, a precondition for the successful application of this type of on-line interactive questionnaire is that users are clearly instructed, i.e. need to understand exactly what they are supposed to do and how they are supposed to respond (Crellin et al., 1990, 333). The analysis of the data arrived at by means of the PROTEUS shell needs a certain experience with the elicitation technique. The system performs computer aided cluster analysis and graphical display of the numerical ratings and textual construct labels used in the questionnaire.

Apart from on-line interactive questionnaires, debriefing questionnaires (debriefing questionnaires can be both paper or on-line) are still used in post-testing interviews after performing a scenario test (Karat, 1990, 353), cf. also B.5.1.1.3. They generally elicit how the performance of the software was judged, what changes need to be made to the software, what was confusing, what was particularly helpful etc. They are often used in combination with post-testing interviews which are performed after the analysis of the questionnaires and which take up those aspects that need further clarification.

B.5.1.1.2 Checklists Checklists are frequently used for manual software tests, particularly for any kind of inspection. Generally the aim of using checklists is "... to obtain a concise and coherent description of the system in terms of objects, attributes, functions, relations between objects as well as between objects and functions, dialogue states, selections and estimated usability" (Vainio-Larsson, 1990, 325).

Checklists can deliver numerical, boolean and classificatory values. There are various rating techniques applied in checklists, the most important of which are availability rating and performance rating (Athappily and Galbreath, 1986, 15–21). The most frequently used rating technique in checklists is availability rating, for which only the boolean values of yes/no (or true/false) are used. A more comprehensive rating score, normally with 5–7 items, is used for performance rating. An often used performance rating score covers the following 5 items (Athappily and Galbreath, 1986, 15) :

0 feature not available
1 feature partially available
2 feature limited or fair
3 feature complete or good
4 feature very good or superior

For quality characteristics such as usability, understandability, learnability, etc. the performance scale would be adjusted, i.e. instead of "available" the scale would take up "usable", "understandable" etc.

Ideally a checklist used for testing purposes should take up the metric that is being applied, the measurement technique, and, most importantly, the related quality characteristic. A sample checklist for the inspection of a Translation Memory that takes up the quality characteristics *inter-operability* and *understandability* is shown in Figure B.8 .

A comprehensive inspection checklist should tackle every quality characteristic that is of importance for the particular test case. A great number of aspects that have to go into an inspection checklist can be derived from the user profile or system specification. Other quality characteristics and metrics, such as many of those for *usability* are of general importance and are thus not the subject of any specification document. They are not application specific and therefore, once elaborated, can be taken up in many inspection checklists equally.

Each checklist must be adapted to the specific test type. Apart from inspection, checklists are needed for observations carried out in the framework of scenario tests (Höge et al., 1991, A-III) and (Höge et al., 1992, A-4). Experience proved that for scenario checklists, it is most adequate to use a table format that has at least the following columns:

- description of sub-tasks
- function performed
- user comments
- observation remarks
- user errors/problems
- help request
- system failure
- time of action

Characteristic	Metric	Measurement	Value
inter-operability	integration of program into text processing	possible?	Y/N
	integration of other data resources (e.g. term bank)	possible?	Y/N
	sharing of databases etc.	possible?	Y/N
	how many databases available?	number	
understandability	sequence of windows/actions easy to understand?	understandable?	0 - 1 - 2 - 3 - 4
	names of buttons/icons/menus easy to understand? etc.	understandable?	0 - 1 - 2 - 3 - 4

Table B.8: Sample Checklist for Translation Memory Inspection

Effective direct observation B.5.1.1.4 depends to a large extent on the suitability of the checklist. Thus the checklist needs to be well organised, providing the possibility to take up every item that relates to those quality characteristics of interest. At the same time a scenario checklist needs to be flexible enough to follow unexpected user behaviour. Whereas it is pretty easy to fill in inspection checklists, effective checklisting in scenario tests is very difficult. This is mostly due to the fact that the observer has to do two things at the same time, i.e. observing and noting. Thus it is advisable to perform pilot observations with a draft checklist before actually entering a test, in case the observer has no checklisting experience and/or the appropriateness of the checklist has not been tested before.

B.5.1.1.3 Interviews Interviews are part of most scenario tests. In an interview one or more interviewees are questioned by one or more interviewer(s). When aiming at the elicitation of very critical, personal data, it is advisable to perform personal interviews involving only one interviewee and one interviewer, whereas for more general data elicitation, a group of interviewees is likely to stimulate the readiness of the individual to provide detailed answers and thus will inevitably increase the amount of data.

When planning an interview, one has to decide *when* and *how* the interview should take place. Concerning the point in time at which interviews are typically performed, one may distinguish between pre- and post-testing interviews. Pre-testing interviews are performed in order to elicit the subjects' personal backgrounds, opinions and expectations concerning the system that is going to be tested (Moll and Ulich, 1988, 73) and (Crellin et al., 1990, 330). The information gained by means of pre-testing interviews gives valuable hints when interpreting the scenario test results. Post-testing interviews are an important, if not necessary, part of each scenario test (Höge et al., 1991, 22) and (Crellin et al., 1990, 330). They are performed after the observational data, i.e. on video tapes or checklists, is analysed. Each aspect that needs further clarification is taken up in the post-testing interview. When performed in conjunction with video observation or think-aloud, the behaviour and comments of the subjects in particular situations can be discussed with the subject and analysed jointly. A combination of both pre- and post-testing interview is particularly useful, since it allows the assessment of the change of mind of subjects during the testing exercise. Cf. (Moll and Ulich, 1988, 73) who reported that at the beginning of a test, attitudes towards the usefulness of help systems were quite positive, while at the end, after having used the help system various times, it was much more negative.

There are various possibilities to perform interviews. A basic distinction which stems from knowledge engineering is between focused and structured interviews (Fulford, Höge and Ahmad, 1990, 14). In a focused interview, the interviewee is prompted with a question related to his/her working environment, i.e. typical tasks, problems etc. and his/her general opinion towards the system under testing. The interviewee is thereafter given the opportunity to express him/herself freely while being interrupted as little as possible (Fulford and Höge, 1989, 16) and (Crellin et al., 1990, 330). "The principal aim of the focused interview is to obtain a typology of objects and agents in the domain, to establish basic factual knowledge, and to achieve a breakdown of the problem" (Fulford and Höge, 1989, 17).

The structured interview is used for obtaining detailed information on specific topics which arose from testing. During the structured interview, the interviewer often follows prepared checklists, hands out debriefing questionnaires which discuss specific topics related to the software (cf. (Karat, 1990, 535) and B.5.1.1.1), shows a series of storyboards to enable the interviewee to visualise possibilities of the screen layout (Fulford et al., 1990, 14), or distributes multiple-choice tests, eliciting, for instance, the understandability of help messages etc (Moll and Ulich, 1988, 73).

Ideally, interviews are audio recorded, which eases later data analysis. The major advantage of interviewing lies in the fact that, unlike e.g. interactive on-line questionnaires and observations, it "...does not interfere with the processes as they take place." (Crellin et al., 1990, 330). Moreover it allows to elicit information on important aspects such as the organisational constraints of integrating the software into the every-day work etc. Also, the post-testing interview is the only means to verify or falsify the interpretations made from data which was collected by the aid of other instruments such as observations or questionnaires and thus is an important means to ease data analysis, since the probability of incorrect data interpretation is decreased. On the other hand, however, the major disadvantage of interviews lies in the fact that the lack of anonymity in personal interviews may drive interviewees to suppress important information willingly. Or, on the part of the interviewer, bias, inexperience or fatigue may distort the data (Crellin et al., 1990, 330), while on the part of the interviewee "post-hoc rationalisation may occur and conceal evidence of the actual processes that took place" (Crellin et al., 1990, 330).

Despite all criticism on the subjectivity of interviews as instrument for data collection, it has to be noted that when performed in combination with other instruments such as, above all, observations and questionnaires, the role of interviews should not be underestimated.

B.5.1.1.4 Observations Observations are the most important instrument in any kind of software test involving users. Observations can deliver results on all user-related quality characteristics such as *understandability*, *learnability*, *operability*, *task adequacy*, *task-relevance*, *usability*, *comprehensibility*, *error-tolerance*, *consistency* etc. as well as on the *functionality* and the *efficiency* of the software system (Oppermann et al., 1988, 11).

One may distinguish between direct and indirect observation. When performing direct observation one or more evaluator(s) sit close to the subject, while watching and taking notes on prepared scenario checklists (Karat, 1990, 330), (Höge et al., 1991, 22), (Höge et al., 1992, 8) and (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10). As pointed out before, the success of direct observation depends to a large extent on the appropriateness of the checklist and the experience of the evaluator. When interpreting the results of direct observations, one has to keep in mind that it is very difficult to observe users without intruding, which often alters the nature of the interaction with the system (Crellin et al., 1990, 330). Thus direct observation should always be combined with post-testing interviews which allow the discussion of user behaviour. However, even if combined with checklists and interviews, direct observation will mainly provide results for those metrics that could be pre-defined, i.e. those aspects to which the observer paid attention and which could be noted on the checklist.

For indirect observation either video recording or one-way mirrors are used. Though video recording is not as intruding as direct observation, it still has a certain effect on the behaviour of the user. Thus, video recording is particularly useful in combination with other test instruments such as think-aloud and interviews. Think-aloud and video allow the development of the user's mental model of the software (Moll and Ulich, 1988, 74). This is particularly interesting, if there is a clash between user expectations and the actual performance of the system, something which is often the case for conceptually new software as in the NLP area. In the TWB project this experience was made concerning the users' attitude towards style checkers, towards which the subjects showed an extremely negative attitude before testing, which was due to their expectations of the system actually criticising their personal style. Not knowing what a computer is capable of doing, users are often disappointed if the functionality of the system is more limited than what they expected.

Post-testing interviews are also useful, particularly when they include the re-play of video segments that posed problems in the data analysis phase. Subjects are then asked to watch their behaviour in "critical" situations and comment on their behaviour and problems encountered. Cf. (Moll and Ulich, 1988, 74) who call this "Selbstbeobachtung aus der frischen Erinnerung".

The major advantage of video recording is that it allows reviewing the data as often as necessary for thorough data analysis. However, viewing the tapes several times, identifying interesting segments that can provide information on those quality characteristics that are important for the particular case, reviewing these segments and capturing the details, is a considerable amount of work (Vainio-Larsson, 1990, 325). Thus while video recording asks for little time in the test preparation phase, the data analysis phase is n times longer than with direct observation (Crellin et al., 1990, 332). Accordingly the expenses that have to be planned when integrating video observation have to consider both additional equipment and high personnel costs.

One-way mirror observation is only used in well-equipped laboratories (Karat, 1990, 353). It is the least obtrusive way of observing subjects. Similar to direct observation, one-way mirror observation has to be combined with a scenario checklist on which to note user behaviour. Concerning data analysis the same is true as for direct observation, i.e. that mostly only those aspects are captured on which the observer has an eye during observation.

Observations need to be performed in conjunction with other forms of data collection, since (i) important "internal" aspects are not available for data interpretation, and (ii) the detailed interaction with the system cannot be completely covered. Complemented by interviews or think-aloud as well as logfile recording, the quality of results increases considerably.

B.5.1.1.5 Think-aloud protocols Think-aloud protocols are used in many empirical investigations. It is a means of qualitative data collection. The motivation behind using think-aloud protocols is to collect information on the users' own reasons for their behaviour (Vainio-Larsson, 1990, 325), (Moll and Ulich, 1988, 74) and (Crellin et al., 1990, 331). The data collected in think-aloud protocols needs to be evaluated carefully, since thinking aloud presupposes that users are able to describe their actions, which is only true for users trained to verbalise their thoughts (Vainio-Larsson, 1990, 325). Also, what users are able to verbalise, represents only the conscious part of their thoughts and thus neglects important subconscious aspects (Hönig and Kusmaul, 1982, 82). Another problem of applying think-aloud protocols is that it may have a negative effect on the user behaviour as it is recorded by direct or indirect observation (Crellin et al., 1990, 330). This is due to the fact that it is even more intrusive than pure observation, (Crellin et al., 1990, 330), and, moreover, "... many users have difficulty in acting and reflecting simultaneously." (Vainio-Larsson, 1990, 325).

Due to the various problems related to think-aloud protocols, in testing practice think-aloud protocols are only used as a complementary method to ease data interpretation. As such they are valuable, as long as the related context and problems are considered in data interpretation.

B.5.1.2 Automatic test instruments for user-oriented testing

Most automatic test instruments are developed for specific types of application, e.g. to perform benchmark tests for translation memory programs and the like. For details on NLP toolkits see (Galliers and Jones, 1993, 129). The aim of this chapter, however, is not to describe individual toolkits but rather to provide the reader with some information on generally usable automatic test instruments.

Those automatic test instruments that can be used with different types of applications in user tests are mostly geared to support data collection. They elicit both qualitative and quantitative data and are useful supplements to manual test instruments in user-oriented software testing. They are additional data collection instruments which are in the first instance rich in data and do not intrude on the user's thoughts or activities (Neal and Simons, 1985, 1052) and (Crellin et al., 1990, 335). They document the actual user behaviour on the system, which, in the analysis phase, can be compared to what the user thinks he/she was doing (Crellin et al., 1990, 331).

There are many different names that denote two major types of tool, i.e. (i) logging programs that time-stamp and record the user-interaction with the system into files that can be printed and analysed after test completion, cf. "interaction logging" in (Vainio-Larsson, 1990, 325) and in (Karat, 1990, 353), "keystroke level model" in (Oppermann et al., 1988, 11) and in (Crellin et al., 1990, 333), "system usage log" in (Crellin et al., 1990, 333), and (ii) playback programs that record user-interaction and provide playback facilities for later analysis, cf. "playback methodology" in (Neal and Simons, 1985, 1052), "logfile recording" in (Moll and Ulich, 1988, 73), "interaction log" in (Crellin et al., 1990, 353). The major difference between the tools, therefore, does not lie in their aim but rather in the way they support the data analysis phase. Whereas the data of logging programs ask for a "manual" analysis of the interaction data, playback programs actually show the whole testing session on a second computer (Neal and Simons, 1985, 1054).

Recording not verbalised operations, i.e. all keystrokes and mouse activities, including incorrect inputs (Vainio-Larsson, 1990, 325), the data provides useful information on quality characteristics related to the *usability* and *functionality* of the software. For instance the frequency of use of a certain function within several testing sessions gives some hints on the *task-relevance* of the function (Moll and Ulich, 1988, 73), (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993,

10) and (Vainio-Larsson, 1990, 325), the occurrence of cumulative handling errors of users provide information on the *understandability* as well as on the *learnability* of the function (Moll and Ulich, 1988, 73) and (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 10). The *suitability* of the help function can be assessed from the number of cases in which, after the consultation of help, solutions were found etc. (Moll and Ulich, 1988, 74).

Logging and playback programs are general data collection programs that are external to the software under testing, i.e. no changes are necessary from one application to another (Neal and Simons, 1985, 1052). It can be used with actual product code or prototypes of the user interface of a product under development (Neal and Simons, 1985, 1052). The application of toolkits for logging programs which are available on the PC market generally requires intimate knowledge of the systems under testing; since the interfaces between the application and the logging program need to be specified (Crellin et al., 1990, 331).

B.5.2 Reporting instruments

Successful testing and evaluation is not only a matter of the choice of metrics and the optimal combination between test type and instrument for the particular test environment, but also includes a detailed, correct and adequate reporting of the test results. Among those reporting instruments that are of relevance for user-oriented testing are

(i). evaluation descriptions cf. "test description" in (Deutsch, 1982, 290) or the notion of "setup" in (Galliers and Jones, 1993, 29–30) and also in (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 14), and "testing environment" in (Höge et al., 1992, 7).

(ii). test problem reports (Deutsch, 1982, 289), cf. "software trouble report" in (Thaller, 1993, 123), "failure sheets" in (Höge et al., 1992, B–3/1–B–3/7).

(iii). result reports, cf. "result sheets" in (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 71–130), "metric-level tally sheets" in (Murine and Carpenter, 1983, 375) ;

(iv). assessment reports, cf. "summary of test results" in (Deutsch, 1982, 290) and in (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 17–26), "criteria summary sheets" in (Murine and Carpenter, 1983, 374).

B.5.2.1 Test descriptions

Test descriptions cover all factors that influence the overall evaluation procedure, including all details that are necessary to judge the performance of the test and to verify the interpretation of results. There are four major factors that determine the type of evaluation and the corresponding testing exercise, i.e. first and foremost, the motivation behind evaluation (Galliers and Jones, 1993, 186), the system and its parameters, the evaluation environment, and, finally, the quality requirements that need to be tested.

Galliers and Sparck-Jones rightly stress the importance of determining the detailed **motivation** behind evaluation (Galliers and Jones, 1993, 141,186). Particularly the three principal aspects that denote the motivation behind evaluation need careful consideration, i.e. perspective, interest and consumer of the evaluation:

(i). the *perspective* of evaluation denotes whether one is interested in the tasks which a system takes over (task-oriented), or the amount of money that can be saved when implementing the system (financial), or how the system can be included into an existing working environment (administrative) etc. (Galliers and Jones, 1993, 186).

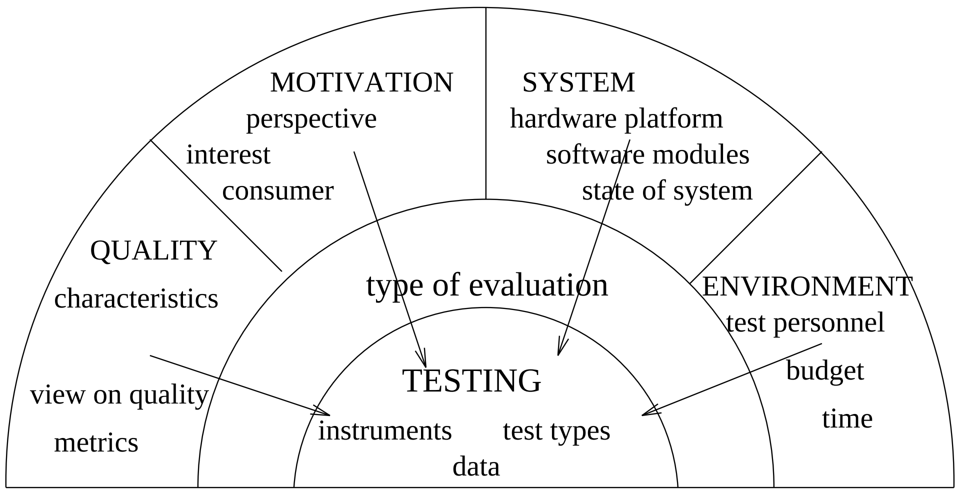


Figure B.2: Factors Determining Evaluation Procedure

(ii). the *interest* (possible interests reported by (Galliers and Jones, 1993, 186) are developer/funder ...) taken in the evaluation denotes the view taken on the evaluation exercise, i.e. even for the same type of evaluation, a developer may have a totally different view on what needs to be considered than the funder of a project, who will again put different foci than user-organisations etc.

(iii). the *consumer* (possible consumers reported by (Galliers and Jones, 1993, 186) are manager/user/researcher ...) of the evaluation report denotes whether managerial, scientific, practical or implementation related aspects are focused during evaluation and reporting (Galliers and Jones, 1993, 141).

The **system** under testing is another factor that needs to be closely considered in an evaluation exercise. The settings of system parameters such as hardware platform, software modules and the state of the system, i.e. whether the system is a prototype, β-version or product, are givens and strongly determine the results of the evaluation exercise. For evaluation purposes it is important to note the difference between external system parameters which normally remain constant during an evaluation exercise and internal system variables, i.e. "properties of or constraints on the system's inputs and outputs", (Galliers and Jones, 1993, 24) the values of which may be changed for evaluation purposes. In the case of a translation memory, for instance, the value for the target fuzzy match percentage may be changed from 90% to 60% which will inevitably lead to more translation proposals. However, whereas changing internal variables will only lead to different evaluation results for the *same* metric, changing the parameters may also lead to the choice of *different* metrics. If, for instance, a system prototype is evaluated, metrics measuring *efficiency* are not likely to be applicable, while for product evaluation, *efficiency* certainly is one of the most important quality characteristics.

The evaluation **environment** is generally determined by the test personnel, the size of the budget and the amount of time invested. Test personnel can be divided into two major groups: (i) experts that function as evaluators during an evaluation exercise and (ii) users that function as subjects of tests (Falkedahl, 1991, 20). On the evaluator side, it is important to note the number of evaluators participating in tests, their function, i.e. whether responsible or assistant, their educational background, and, last but not least, their experience with the test types and instruments used. User-oriented testing should preferably be performed by someone who was not involved in the development of the software (Thaller, 1994, 61,90) and (Hausen and Müllerburg, 1982, 119). A certain knowledge of software engineering principles, and of both the practical and the computational side of the application under testing is, however, very useful, if not indispensable.

On the subject side it is particularly important to note how many and what type of users participated in tests. Depending on the quality characteristics and the metrics applied, as well as on the test type and instruments, different types of user - students, professional users - can or rather need to be used as subjects (Falkedahl, 1991, 21). The crucial qualifications required by test persons are (i) objectivity, (ii) representativity, and for all NLP applications (iii) languages required (Falkedahl, 1991, 21). Thus for each individual user the above three qualifications have to be checked in the test planning phase and the background noted in the test descriptions. The quest for objectivity is particularly difficult in the NLP area, where there are rather few people who have some idea of the functionality of the systems, but at the same time do not have previous experience with one system or the other, and consequently might judge the system under testing with respect to the similarity with the systems they know. “It must therefore be noted that striving for objectivity runs the risk of making evaluations and tests using purportedly unbiased evaluators and test persons into purely artificial events whose results are likely to be insufficiently informative about or representative of how a system would be judged by its end-users, once they had become accustomed to its peculiarities.” (Falkedahl, 1991, 22).

The evaluation budget is naturally the most decisive factor, when it comes to selecting evaluators, subjects, test types and instruments as well as when determining the time that can be invested into evaluation. In professional evaluation environments, planned and actual evaluation costs need to be calculated and noted on the test description. It is important to note here that, in case of a limited evaluation budget, it is advisable to reduce the number of metrics that will be tested and to select less expensive instruments, rather than to reduce the number and qualification of test personnel. While a limited number of metrics only reduces the scope of the evaluation exercise, savings in the area of test personnel lead to less reliable test results.

The notion of software quality cannot be defined in general terms. It is a function mainly of perspective, interest and consumer of the evaluation exercise. While, for instance, from a financial perspective, *efficiency* is the most important quality characteristic, an administrative perspective will rather focus on *inter-operability* and *usability*. Both consumer and, in particular, the interest behind the evaluation exercise are responsible for the view on quality that is being taken during evaluation. The glass box view on quality distinguishes between three dimensions, i.e. (i) data, (ii) system - interface, and (iii) system function, while the black box view only distinguishes between data and system dimension, neglecting the fact whether a certain result goes back to the system's interface or its functions. For instance, while a developer will be interested in the more differentiated glass box view on quality, the funder of a project will be satisfied with the less complex black box view. In practical terms, it will be enough information for the funder of a project to learn that the quality characteristic of *understandability* was judged low in a termbank, because e.g. the definitions were not understandable (data dimension), and the system as such was difficult to grasp (system dimension), whereas the developer additionally needs to know whether it was the windowing sequence that caused problems (system - interface dimension), or the problems rather stem from the implementation of too complex operations (system - function dimension). The type of metrics, i.e. whether quantitative or qualitative, depends to a great extent on perspective and, above all, the consumer of the evaluation. Even for the same quality characteristic, e.g. *reliability*, a scientist is likely to perform mathematical stress and volume tests, leading to quantitative values, while when aiming at the final user of the system, evaluation might rather cover the application of qualitative questionnaires. For details on stress and volume tests see B.3.2.1.

In addition to the four factors determining the setup of an evaluation exercise - motivation, system, environment and quality - the three types of evaluation as defined during the Edinburgh evaluation workshop of 1992 (Thompson, 1992) give some more general insights into the overall test bed and are therefore useful to be noted in the test descriptions. If, for instance, a software system performs badly in the β-version and one has to find out why, the type of evaluation performed is *diagnostic* evaluation. If one has to find out to which extent a new version of a software system performs better than the old version, the type of evaluation is *progress* evaluation. Finally, if the aim is to find out whether the software system is suitable for a particular environment, the type of evaluation performed is *adequacy* evaluation. It must, however, be noted that the three types of evaluation, though conceptually clear, can occur in any combination during a practical evaluation exercise.

The final evaluation description document covers all of the above aspects, including a short description of test type, instruments and data used. Copies of the actual test instruments and data used for the tests have to be added as a separate appendix.

B.5.2.2 Test problem reports

The major usage of test problem reports is in the framework of software development projects rather than with off-the-shelf products. Thus they are mostly relevant for diagnostic and progress evaluation rather than for those adequacy evaluation environments in which no feedback between evaluator and developer is planned or possible.

Test problem reports provide developers with the detailed description of problems that occur during testing. They are very important instruments that aim at the improvement of software under development. The most important part of test problem reports is the detailed description of the problem and the actions that led to the problem (Thaller, 1993, 123), (Deutsch, 1982, 289) and (Höge et al., 1992, B–3/1—B–3/7). In diagnostic evaluation environments a diagnosis of the failure is given and the action required is described, if possible (Thaller, 1993, 123) and (Deutsch, 1982, 289). For calculations such as MTTF (mean time to failure) or MTBF (mean time between failures) it is necessary to put down the exact time when a failure occurred. Another important aspect that needs to be noted in test problem reports is the priority ID of the failure. Figure B.9 shows a representative failure priority score presented by Deutsch (Deutsch, 1982, 289).

1	fix immediately - catastrophic error, test cannot proceed
2	fix before test completion - serious error, severe degradation in performance, but can continue test process
3	fix before system acceptance - moderate error, specification can be met
4	fix by a specific date or event
5	hold for later disposition
T	nonrepeatable occurrence - problem will be tracked for reoccurrence
X	new problem - problem assumed to be serious but insufficient data available for analysis, investigation required

Table B.9: Failure Priority ID Score

In general test problem reports are an important part of the auditing of software projects and therefore have to take up every aspect that is needed for correct auditing and time-management (Thaller, 1993, 123).

B.5.2.3 Result reports

While for feature inspection, the results are reported directly on the feature checklists used during inspection, result reports integrate all results achieved by means of scenario tests and systematic testing. They cover all details on the metrics applied and observations made during the different testing exercises and are normally provided as appendix to the overall test documentation. On the contents of result reports see also (Oppermann et al., 1988, 8) and (Murine and Carpenter, 1983, 374). It is a means that allows interested parties to look up the detailed results of the tests.

Testing experience in the TWB projects led to an optimised design of a result report as it can be used for the documentation of any test environment that includes scenario and systematic testing (Höge, Hohmann, van der Horst, Evans and Caeyers, 1993, 71–130). Depending on the type of evaluation performed, i.e. whether progress, diagnostic or adequacy evaluation, the result reports need to be more or less comprehensive.

TYPE OF INFORMATION	ITEM OF INFORMATION	TYPE OF EVALUATION
administrative	system - name - version - vendor	adequacy progress diagnostic
	evaluator - name - affiliation	
	date	
	test type	
evaluation	function	adequacy
	observation/metric	diagnostic
	result description	progress
	related quality characteristic	
evaluation	proposal for improvement	diagnostic
	dimension of quality	progress
	priority	
	developer remarks	
	deadline for implementation	

Table B.10: Information Categories in Result Reports

While administrative information is relevant for all types of evaluation, adequacy evaluation requires a minimum of evaluation information, including the function that is discussed, the observation made or metric applied, the description of the results of the observation or metric, and, finally, the related quality characteristic. In addition to the above information, in diagnostic and progress evaluation it is important to note to which dimension of the system the result relates, i.e. data, system-function or system-interface, if possible, to give a proposal for the improvement of the software, how important it is for the evaluation side that the improvement be taken up, how the developers reacted on the results of the test and the proposals for improvement, and, finally, by when developers plan the implementation of the changes.

In case the result report is used in the framework of software development projects, additional information for the auditing and time-management of the projects is necessary, i.e. whether implementation deadlines were met or not, and if not why not.

B.5.2.4 Assessment reports

Assessment reports are top-level reporting instruments that are based on the detailed data which is provided as appendix, i.e. result reports, test descriptions, and test problem reports. The testing of a software system leads to a great number of individual results that are documented mainly on the metric level. In order to gain a picture of the whole software performance, it is necessary to proceed from the specific result on the metric level over the respective sub-characteristic to a general statement on the top level of quality reporting, i.e. evaluating the system's performance in terms of its eventual *functionality*, *reliability*, *usability*, *efficiency*, *maintainability*, and *portability*. Cf. (ISO, 1991).

(Murine and Carpenter, 1983, 374) point out that "All observed and measured data is recorded at the metric level; scores at the criteria and factor level are generated therefrom." Evaluation of test results implies the comparison between the actual software performance and the pre-defined target quality standard (Höge et al., 1991, 2) and (Schmied and Winkler, 1989, 6–9). Thus, the results on the metric level have to be considered in the light of the stated or implied requirements of the respective user group. However, in user-oriented testing it is not always possible to define the target quality in exact terms. This is mostly due to the facts that (i) in user-oriented testing there is an obvious lack of quantitatively measurable metrics, and, (ii) the capability of individual functions cannot always be anticipated by a testing team which normally does not include system developers. In the NLP area, for instance, it is often not clear what systems such as translation memories or machine translation actually can do, and what the best solution could look like.

Also, for each individual evaluation scenario, different focus is put on the quality characteristics. If, for instance, a tool for object-oriented programming is being evaluated, there is a clear focus on *functionality*, *maintainability* and *portability*, while for software developers as the target user group of the tool, *usability* is of minor importance. Or, if a remote batch machine translation program is being evaluated, its *functionality* is much more important than its *usability*, since there is only little interaction between system and users.

Another important aspect that has to be considered when evaluating the final performance of a software system, particularly of NLP applications, is the dimension of the system that is being considered. While for the black box view on quality, which is mainly used for adequacy evaluation, it is only relevant to distinguish between the system and data dimension, diagnostic evaluation asks for a glass box view on quality which further distinguishes on the system side between the function and interface dimension. The *reliability* of a translation memory, for instance, needs to be considered on the data dimension, i.e. whether the system actually recovers the nearest translation, whether it provides data on the originator and the date of the translation etc., and on the system dimension, i.e. whether there are frequent system breakdowns etc. It may well be that a translation memory is very reliable on the system dimension but on the data dimension, for instance, does not provide the originator of the translator and fails to offer the closest translation proposal.

B.6 Conclusion and outlook

This appendix is a selection of methods for the measurement of software that can be applied to any software system, but it was compiled in view of the special problems that software evaluation poses in the NLP area. As has been repeatedly pointed out, each evaluation scenario asks for a particular combination of test types and instruments just as each environment asks for the measurement of different metrics. Thus this report has to be understood as practical help and guideline for the testing of NLP software, as it fairly exhaustively discusses the possibilities at hand, their merits and drawbacks.

It is suggested that working groups within EAGLES evaluation assess the different test types and instruments and test the practical guidelines delivered in this appendix. Although it is based on practical evaluation experience with NLP applications, further practical testing exercises will certainly lead to a useful discussion and further development of the testing framework as it is presented here.

A long term objective of the EAGLES evaluation group is the standardisation of evaluation procedures. Here test types and instruments are among the prime candidates for standardization. It is hoped that this report will contribute to this effort.

Appendix C

Requirements Analysis for Linguistic Engineering Evaluation

C.1 Introduction

This report describes some investigations into issues and methods for user profiling and requirements analysis for Linguistic Engineering (LE) evaluation. In part it is a fuller version of the section on requirements in the main body of this report (Section 2.4).

The input to the ISO process model of evaluation (2.1.1) is a statement of **stated or implied needs**; part of the preparation phase of evaluation consists of expressing these needs as valid and measurable attributes of system performance (see section 2.3). The work described here is relevant to the process of arriving at such statements of need (or **requirements statements**) and validly transforming them into suitable attribute specifications for systems. It concentrates on **adequacy evaluation**, i.e., the evaluation of the fitness for purpose to some user of an existing software system. Many of the points carry over into other sorts of evaluation, however. The examples that are used are primarily taken from the Writers' Aids area, concentrating on spelling checkers and grammar checkers.

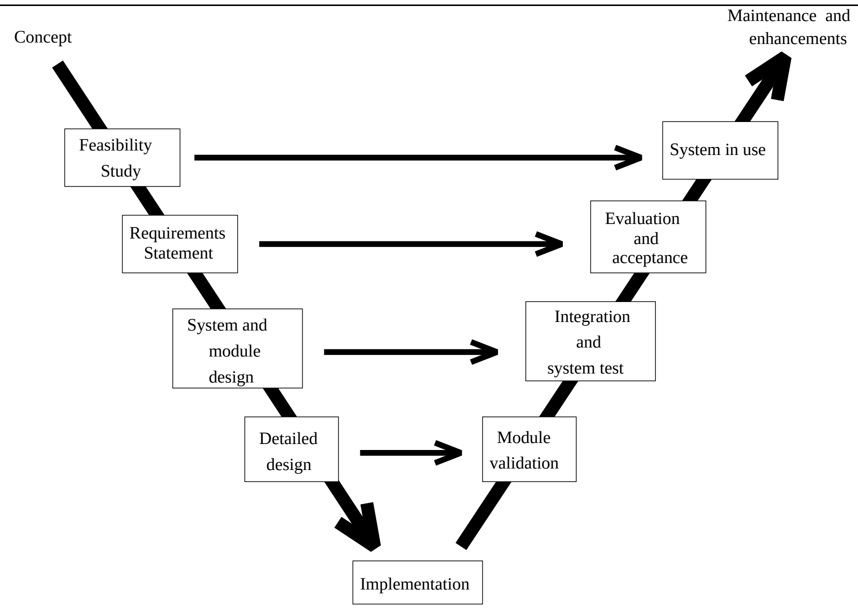
This work cannot be seen as the final word on requirements analysis and user profiling for linguistic engineering evaluation. It is effectively the work of one person, and has yet to be subjected to detailed review and use by other Evaluation group members. Because of the timescales, it has not been possible to make all the subgroup reports of the EAGLES Evaluation Report conform to the method outlined here. It is hoped that the development of generic task models and associated user profiles, and the development and testing of knowledge acquisition techniques for the various kinds of information that go into them, will be continued — the work is necessarily cumulative.

C.2 Requirements analysis and evaluation

In this section, some background is given for the relationship between requirements analysis and evaluation in general software engineering. Some terminology is introduced, and the relationship between requirements analysis and user profiling clarified.

C.2.1 Evaluation in the software life cycle

In most accounts of the Software Engineering life cycle, evaluation is carried out in terms of the **requirements** elaborated in the first phase of the software development process: **Requirements Analysis**. More detailed, more committed descriptions of the system, which fall under the heading



(adapted from Tansley and Hayball (1993))

Figure C.1: 'V-diagram' showing the place of evaluation in software development.

of design, are used to test software modules. Figure C.2.1 shows how the outputs of the various design stages are typically fed into the evaluation process.

Requirements engineering is a growing field of enquiry in software engineering, and it is intend to use the products of research in requirements engineering and in the general software engineering field in this work in so far as they are useful. However, linguistic engineering as an application area, and the particular purposes of the different sorts of evaluation distinguished by EAGLES, have special characteristics, and require the development of special-purpose versions of requirements procedures. This is in line with recommendations in the requirements engineering literature, and later in this report will be illustrated by the example of the KADS methodology for KBS.

C.2.2 Scope of requirements statements

The different levels of analysis and design can be thought of as different **descriptions** of a problem and its possible solutions.

The relevant space of descriptions has been described Jackson (1995) as covering two intersecting sets of attributes. The set $\{Di\}$ is a set of attributes of the **problem domain**, stated in terms of that domain without reference to system design decisions; the requirements statement proper is a set of relations on D, or constraints involving terms from D.

The set M is a set of **machine** attributes. The intersection of D and M, called S, is, in Jackson's terms, the area of **specifications**, where attributes exist in both the problem domain and the system design, and where constraints derived from problem domain requirements are expressed in terms of machine requirements. (In terms of the Consumer Report Paradigm of our framework, specifications are constraints on the values of reportable attributes.) Figure C.2.2 illustrates the intersection.

C.2.3 Functional and non-functional requirements

In general, requirements are partitioned into **functional requirements** and **non-functional requirements**. Functional requirements are associated with specific functions, tasks or behaviours

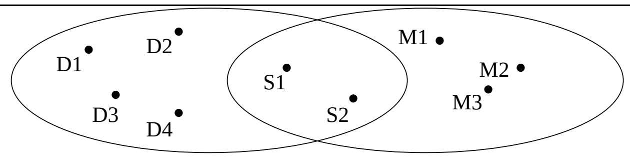


Figure C.2: Domain and machine attributes and specifications

the system must support, while non-functional requirements are constraints on various attributes of these functions or tasks. In terms of the ISO quality characteristics for evaluation (2.1.1), the functional requirements address the quality characteristic of functionality while the other quality characteristics are concerned with various kinds of non-functional requirements. Because non-functional requirements tend to be stated in terms of **constraints** on the results of tasks which are given as functional requirements (e.g., constraints on the speed or efficiency of a given task), a task-based functional requirements statement is a useful skeleton upon which to construct a complete requirements statement. That is the approach taken in this work. It can be helpful to think of non-functional requirements as adverbially related to tasks or functional requirements: how fast, how efficiently, how safely, etc., is a particular task carried out by a particular system?

C.2.4 User profiling and requirements analysis

User profiles behave like **parameterisations** of requirements statements, capturing regular variation in requirements for similar types of system.

Different users may have different functional requirements, and so require different subsets of functionality to be evaluated, or they may have different non-functional constraints on functions.

The terms "user" and "user profile" will not be used so generally in what follows. It is the terminology used in our efforts towards formalisation(2.2); however, the factors that affect and parameterise software requirements do not only derive from end-users of the software, nor yet from only human elements in the software's intended environment.

The term that will generally be used in what follows for the overall context of use is **setup**, following Galliers and Jones (1993); this will be returned to later. Within that usage, the actual operator of software will usually be called the **end-user**; the person or organisation to whom the evaluation is addressed will be called the **customer** of the evaluation. Profiling, or the systematic acquisition of information about variable requirements, can be applied more or less independently to many different parts of a setup. Particularly where we are talking about adequacy evaluation, it may be necessary to analyse a number of different setups to capture the (possibly disjoint) requirements of all customers of the evaluation.

C.3 Issues for requirements analysis

There are a number of issues which are important for LE evaluation which any requirements analysis method must take into account. It is rather difficult to disentangle them, and the following sections are clearly not the only way to arrange such a discussion.

In many of these areas, it seems likely that if LE evaluation can make any headway, it will be contributing as much to the state of the art in Requirements Engineering as *vice versa*. What follows does not address all these issues adequately by any means, but they should be borne in mind for further work.

C.3.1 Reporting evaluations to the customer

The main purpose of a requirements statement from the point of view of software development is to support the subsequent design and implementation phases. When our aim is evaluation, this is no longer a priority. A new end-audience for the statement is important, however: the customer of the evaluation. In fact, this is not so different from the classical view; one of the most talked-of issues in requirements representation is the need to re-present requirements for agreement to the management, the commissioning organisation, experts whose knowledge is being represented, etc. (this is one of the biggest difficulties with formal methods).

The Consumer Report Paradigm (3.1) mostly assumes a certain familiarity with the products under test on the part of the customer of the evaluation. This allows the reporting of attributes which relate fairly directly to measurements of the system, and whose relevance to individual customers' needs is taken to be evident. Nonetheless, such attribute listings are usually accompanied by explanatory text which can help to make clear the relationship between characteristics of a customer's requirements that are easy to recognise (because they are couched in the customer's own problem vocabulary) and the attributes whose values relate to those requirements. For Linguistic Engineering, the factors of technical performance are not well understood by any mass market audience, and indeed there may not be an adequate vocabulary for the linguistic facts of the problem domain, regardless of system considerations. Explanation of the relevance of attributes might therefore be considered an important part of the function of an evaluation in the area of LE.

How wide should the external bounds of the evaluation be set? One of the criticisms levelled at consumer reports is that they take a given user task, but then assume a certain product class as the candidates to satisfy that need (taking a restricted view of the task). Innovative consumer reports sometimes take seriously the idea of taking the user's need as primary, and preparing evaluation criteria at the top level which do not take for granted a particular type of satisfier for the need. For instance, instead of evaluating cars as though they were a given natural class of satisfiers, some reports have evaluated cars along with combinations of cycling, public transport, and walking, even choice of living area, as potential satisfiers of users' mobility or access requirements. In the case of spelling checkers, it must be relevant to the impact of spelling checkers on the overall task of document quality assurance to know that the errors that spelling checkers are able to spot are also those that people find it easiest to spot (namely, errors that do not result in other legal but unintended words) (Cohen, 1980) .

If we take seriously the needs of the customer, then, we need to analyse their requirements for explanation, as well as just technical requirements.

In knowledge acquisition for expert systems, it has often been remarked that once you try to get information from more than one 'expert', you are in trouble, even if these are supposed to be experts in the same field, playing the same rôle in the situation being analysed. In fact, the gathering of requirements is complicated by the fact not only of different rôle-players, but multiple rôles in the situation, which may result in completely different **viewpoints** on the problem. These may simply involve different vocabularies to describe compatible requirements, but they may also involve genuine clashes in requirements between different stake-holders, e.g., management and end-users, writers and editors. There is a good deal of work being done in Requirements Engineering on how to represent and manage such multiple viewpoints, some of which may be relevant to our needs (Nuseibeh, Kramer and Finkelstein, 1994) .

Where multiple viewpoints do not involve clashes, they can once again be seen as a special case of the redescription activity which is a fundamental way of conceptualising the requirements analysis process.

In a sense, to explain the relationship between descriptions of requirements that are comprehensible to the customer or audience of an evaluation, and those attributes of systems that can be directly measured is to explain why the system-level attributes are valid measures of customer-level requirements, and hence is related to work in the traceability of requirements and issues about the validity of requirements and measures, which are discussed in the following section.

C.3.2 Context and validity in LE evaluation

As reported in section C.2, it is standard practice to restrict a definite part of the requirements statement to be expressed purely in terms of problem domain attributes, without prior committment to any particular system's means of solving the problem. This is particularly useful as a starting point for an evaluation which will be applicable to multiple, already existing software systems, as is typically the case for adequacy evaluations according to the consumer report paradigm2.1.1. This is because such a statement forms a base-line against which different ways of fulfilling these requirements might be compared and evaluated. It is also likely that this level of requirements statement will also be the most valid, since it is most directly expressed in terms of users' needs.

However, there remains the problem of how to transform requirements at the level of the problem domain into a form in which it is possible to test them against all the relevant systems under test (i.e., to transform them into specifications, in Jackson's terms, or reportable attributes, in our terms). This clearly involves not just top-down development of requirements, but is affected by the nature of the systems under test. In terms of the V-diagram, the downward direction of the process of software development is equivalent to left-to-right progress through the sets in Figure C.2.2, from problem through specification to design in terms of implementable primitives. Instead of aiming at a specification expressed in terms of implementable primitives, the evaluation process aims at identifying a set of reportable attributes expressed in terms of **measurable** primitives, but otherwise the processes are similar in that a bottom-up aspect is imposed by the nature of the available primitives. The validity of the specification or reportable attributes depends on the validity of the problem domain requirements and the validity of the process by which the specification or reportable attributes are derived from them.

Different systems under test may have different ways of scoping the problem they deal with, and hence be difficult to evaluate against one another. Not only that, but where a set of products do appear to converge on a definition of functionality, there is no guarantee that this definition matches user and task-based requirements. These have to be established independently by the evaluator. (Particularly for mass-market software applications, there are commercial pressures towards convergence on the latest 'check-list' of functionalities which are largely independent of utility. It must be part of the evaluator's job not merely to list these features and report on their presence or absence in given products, but to provide judgements about their relation to utility.)

Then again, in the interests of software reuse, a component-based approach is popular in both general software engineering and LE. Various types of progress evaluation rely on the ability to devise a modular breakdown of the overall functionality such that individual components can be specified, evaluated and chosen independently (or designed, implemented and evaluated independently.) However, it can be difficult to compare components against one another when they do not fit in the same way, requiring different setups in order to work at all. As discussed in Galliers and Jones (1993), NLP is an area in which existing systems are seldom suitable for immediate use in a new application area. Any new use will involve more or less customisation of a generic system, for example by lexicon and grammar modification. How then is it possible to evaluate the relative suitability of different generic systems for a task?

It can also be difficult to isolate the system's task for evaluation in some NLP systems which provide interactive and partial support for some user task, such as grammar checkers in the task of proof-reading, because the most valid measurable results (e.g., the quality of the output text) are only applicable to a setup that combines user and system performance. System performance can only be evaluated by factoring out as a separate performance factor the effect of different end-user types on the overall performance. Relative to this independent variable, more detailed, measurable attributes of the system's contribution can then be defined.

Transforming requirements from problem domain terms into reportable attributes can be seen as a type of redescription problem, constrained by the necessity that redescriptions be valid in some sense. In terms of the EAGLES evaluation framework, we need to know where a certain reportable attribute requirement comes from in terms of the actual problem domain, and whether the derivation is valid; information supporting the choice of the attributes used should ideally be part of evaluation documentation, just as much as information about the methods used to arrive at values for the attributes. For requirements engineering in general, such validation is required to show that specifications are equivalent in some sense to the problem domain requirements they are intended to express — that they are in a sense alternative descriptions of 'the same thing'. There is considerable work in Requirements Engineering on the traceability of requirements in this process of redescription or transformation (Gotel and Finkelstein, 1993) and this may be of use in the future to our evaluation work.

However, unlike software design, where the ostensible aim is to produce a design that is fully equivalent to the requirements, part of the purpose of evaluation is to informatively point out where designs fail to fill requirements. As an example, no spelling checkers of the normal type can correct errors stemming from simple typing mistakes which result in legal though unintended words, such as typing *form* instead of *from*; yet a realistic problem domain requirement would certainly view this as a spelling error. The discrepancy arises at the point of defining the idea of a spelling error at the attribute level as a 'best-equivalent' to its definition at the problem domain level; an explicit and structured process of decomposing or transforming problem domain requirements into measurable attributes can provide opportunities for noting where and how such discrepancies occur. While it might be a bit over the top to have a whole attribute devoted to reporting this failure for every system under test, such discrepancies between a problem domain requirement (correct typos) and the available means to satisfy it should be included in any accompanying discussion or guide to the use of the attribute grid.

Ascertaining which factors are truly valid determinants of software quality is a similar problem; there needs to be a program of validation of requirements as well as validation of how well the reportable attributes and the measures and methods associated with them reflect requirements. This process cannot be completely codified, and must to a large extent be driven by open-ended evaluation in realistic situations, feeding back unforeseeable insights into the requirements statement. However, there are a number of knowledge acquisition methods which provide useful frameworks for the generation of descriptions of the setups, and hypotheses about what factors in the setup may be performance factors. These must then be tested by holding other factors constant; the combinatoric testing task may be considerable, but is the only way to isolate the effect of the system under test.

The next section looks at some knowledge acquisition methods that may be relevant to LE evaluation.

C.4 Knowledge acquisition for requirements analysis

C.4.1 Introduction

Until recently, the requirements analysis part of the software development process had been rather left out in advances in the rest of software engineering. Many software systems never get to the stage of customer acceptance and prolonged use, and faulty or fatally limited requirements analysis is often the problem.[1]

New specialist research in requirements engineering is making common ground with knowledge acquisition techniques developed in Knowledge Based Systems/Expert Systems work. This addresses the problem of getting domain expertise (whether of 'experts' or not) into representations usable

[1] E.g., Roman (1985). It is indicative of the increasing realisation of this that Michael Jackson, of Jackson Structured Design, now works in the area of Requirements Engineering.

for subsequent design (or in our case, evaluation). NLP has its own methods, particularly for actual language analysis, and it might well be fruitful to treat these as analagous to experts in KBS design; however, the actual requirements for software systems must take into account more than the basic NLP performance of a system, and concentrate on the user task, where the best practice should be followed with more general methods.

One difficulty for adequacy evaluation (or indeed the design of mass-market rather than bespoke software systems) is that the needs of the user must be taken to be **implied**, since there are no extensionally defined users to state them. It is here that the requirements analysis process will be most different for adequacy evaluation as opposed to progress evaluation, since realistic requirements analysis for design usually requires dialogue with all stake-holders, prototypes and mockups to iteratively develop and test a description of the user's needs.

The following sections give a quick overview of KADS and related knowledge acquisition methods.

C.4.2 KADS

Originating in ESPRIT projects, KADS (Hayward, Breuker and Wielinga, 1987) is widely used within the EU and increasingly beyond as a practical KBS development methodology.[2]

KBS development according to KADS is essentially a process of **modelling** at various levels. Relating this to the ideas of Requirements Engineering, KADS provides a framework of representations and process suggestions for producing system descriptions at different levels of abstraction.

Three levels of analysis are distinguished. At the **Process Level**, a **Process Model** identifies the tasks involved in the domain, data flows and stores, and the assignment of ownership of tasks and data stores to agents. At the **System Level**, the **Co-operation Model** describes in detail the interactions between the system and external agents, and how 'internal agents' (components of the system) will interact; there may also be separate a User Task Model and System Task Model. The third level, the **Expertise Level** corresponds to an **Expertise Model**, which divides the task of describing the expertise level of the system into a number of **layers**:

The domain layer. This is static knowledge describing concepts, relations, and structures in the domain.

The inference layer. The domain layer is described in terms of the different types of inferences that can be made.

The task layer. Knowledge is provided about how to apply the knowledge in the two lower layers in problem-solving activities in the domain.

The strategic layer. Knowledge on how to select appropriate tasks for problem-solving during plan execution.

Various diagrammatic representations are used for many of these models, and these have been found to be among the most useful parts of the approach for promoting reuse.

A major advantage of the KADS approach is in the idea of **generic task models** (hereafter GTMs), also known as **interpretation models**. These are partially instantiated skeleton models for typical tasks or task fragments — such as "classification" or "system diagnosis" (Tansley and Hayball, 1993) — which are stored in **generic task libraries**, promoting reuse and standardisation. Knowledge engineers can use suitably chosen generic task models to guide the knowledge acquisition process in a new domain, refining and combining GTMs to produce a fully specified model.

[2]KADS may have originally started life as an acronym, but uncertainty appears to have developed about what it is an acronym of, and it is generally used as a proper name. 'Knowledge' and 'Acquisition' are definately in there somewhere.

The basic approach of identifying a sequence of modelling steps and supporting the modelling activity with reusable elements is echoed in the proposals made later in this document. KADS has been suggested before as a method for KA for NLP (Mikheev and Moens, 1994) , in which the descriptions of the expertise level were used to structure NLP knowledge acquisition. The proposals made in the main part of this report draw on the whole KADS process as a useful model for the requirements modelling stage of NLP evaluation.

KADS does have weaknesses, which any attempt to adapt it must be wary of. For instance:

- It is difficult to translate between or connect the different layers.
- No-one really uses all the layers; most people really use the diagrams, but the diagrams are not an expressive enough representation for all requirements, e.g., they don't express recursion.
- KADS systems typically end up with huge amounts of documentation for relatively modest systems and are hard to change.

KADS does not itself specify the representation types in terms of which its various models are to be couched. The field of Requirements Engineering encompasses representations ranging from long discursive reports associated with ethnomethodological approaches, to formal languages like Z. A relatively common representation which might be seen as intermediate between these approaches is conceptual graphs (Mikheev and Moens, 1994) — it is sometimes proposed on the grounds that it is more acceptable to users than stricter logical formalisms, particularly since it can be given a graphical representation.

The last point is important, since we must decide what our needs are for representations. Suitable representations must have a two-sided functionality: on one hand, it must be able to express the language of our testing techniques, and on the other it must be able to describe systems in such a way that they are **recognisable** to those who must contribute to the development of evaluation models.

Strictly formal specification languages are not likely to be very usable for large, real, complex NLP systems and their users. However, in the absence of such methods, the organisation and procedure of requirements development needs to be particularly disciplined and thorough, with lots of check-lists to prompt the preparer of the evaluation to obtain relevant information.

C.4.3 Knowledge acquisition (KA) techniques

Many of the various knowledge elicitation techniques which have become established in KBS development are relevant to our more specific task in NLP evaluation. This is particularly true when considering the presentation of material to end-users, when their natural categorisations of phenomena (rather than the categorisations of linguists or computer scientists) must be determined. It may also, however, be true for experts such as the linguists and computer scientists who build the kind of systems under test — establishing their understanding of difficult areas where systems are weak for particular reasons of the techniques used and the phenomena. Such 'inside' knowledge can support the development of 'trick' tests which may serve to diagnose the kind of techniques being used in a system, which in turn may predict a whole aspect of its performance on user requirements.

Organised knowledge acquisition methods assist in the standardisation of the requirements they result in, in the sense that a specific method can make it easier to check or repeat the knowledge acquisition process. An extremely useful survey of knowledge acquisition methods and procedures, with suggestions about circumstances in which different methods are useful, is given in Cordingley (1989).

Methods range from informal techniques such as 'user observation' through common social science methods such as interviews, questionnaires, and discourse analysis (Ericson and Simon, 1984) to more formal techniques used in KA for KBS. The latter are less well known in general, and are briefly introduced here.

The identification of relevant phenomena requires open-ended techniques, perhaps using some sort of scenario walkthrough technique with users. The first[3] step is to identify suitable information providers and a core idea of the tasks of interest. 'Natural' observation may provide information about typical situations and tasks, but it may also be relevant to deliberately elicit information from informants about various common types of situation, as well as rare but important situations, and walk through these to gain further information, a sort of scenario-based requirements elicitation process. At the early stages, it is crucial not to have a ready-made list of categories into which users' views of the domain must be fitted; this can distort the information and prevent the uncovering of relevant phenomena.

As relevant phenomena are identified, it is necessary to categorise them, although the two activities cannot be separated so cleanly. A number of relatively formal methods are in use (e.g., repertory grid analysis, laddering, card sorts — see Cordingley (1989) for details), largely based around the ideas of Personal Construct Theory (Kelly, 1955) , providing constrained sorting or organising operations to allow the development of categorisation schemes based on the identification of contrastive discriminations whose labels are the constructs. In card sorts, for instance, some small and managable set of elements are chosen, and the subject sorts like with like in as many ways as she can think of; each such distinction can be associated with a labelled construct. For instance, a set of spelling checkers might be sorted on the basis of whether they allow sharing of user-defined dictionaries; this would then become a construct. In the use of this sort of technique for product design or evaluation, it is common to include in the list of items to be sorted some 'ideal' product, to facilitate the production of constructs which do not correspond to any discriminations between existing systems, but which are guides to real user needs or wishes.

Constructs, as used, for example, in repertory grids, look very much like reportable attributes; a typical repertory grid based on elements which are spelling checkers might be represented as a matrix with one axis listing different spelling checkers and the other various attributes of the checkers, so that the values in the grid characterise the checkers. The main purpose of the grid, however, is not to evaluate the checkers but to elicit constructs/attributes that validly represent the important discriminations to be made among checkers. The use of an automated software system for representing repertory grids allows the values to be used in statistical analysis to identify clusters of constructs, and compare categorisations by a number of subjects. The labels used with personal constructs are a matter of personal usefulness, and so the process of moving from a set of personal constructs to a sharable set of attributes suitable a wider audience may involve negotiation; there has been some work done on techniques for the identification of clashes in terminology and usage (Shaw and Gaines, 1989) .

There are a number of software systems available which can automate elicitation and analysis using repertory grids, such as WebGrid (Gaines and Shaw, `http://tiger.cpsc.ucalgary.ca/KAW`); Gaines and Shaw (1993).

Corpus analysis remains the main distinctive method in NLP evaluation. Corpus analysis is a recognised part of KA, where protocols of interviews, expert think-alouds, etc., form the documents — there are even attempts to extract domain models, etc., automatically from the transcripts of experts. In these cases, however, the documents are a secondary object of analysis; in NLP analysis, they are often the primary one. In fact, while NLP suffers in comparison with some traditional SE domains in the fact that the input and output of its systems can't be easily and concisely described, it has the advantage that extensional descriptions of required behaviour (instances of

[3] The stage of requirements definition in a software project is never as self-contained as implied by the V-diagram in Figure C.2.1. The V-diagram should only be taken as indicative of the relationships between the *results* of the various analysis and design, implementation and evaluation stages; the actual processes by which these results are arrived at are more likely to follow an iterative procedure, such as formalised in Boehm's (Boehm, 1988) spiral software development process model. Notions of 'first' and 'next' should be understood in this light.

linguistic input and output) can be relatively readily obtained, and constitute objective evidence. The effort involved in analysing this evidence may be considerable, but the effort in collecting and making available suitably prepared corpora is greater. However, it is the only way to provide reliable information about the linguistic nature of the problem domain, as long as the documents are in fact representative of the problem domain[4]

Corpus analysis can be usefully linked to some of the techniques described above, since the presence of a corpus does not determine how to identify or categorise the phenomena represented there. Paying attention to who has categorised the phenomena in a corpus, and what specific methods if any were used, should be useful.

C.5 Requirements analysis in LE evaluation

C.5.1 Introduction

In this section, some rather tentative proposals for requirements analysis for LE are made. The proposals are illustrated using the example of spelling checkers, drawing on work from the LRE project TEMAA (cf. section 3.2.1). (In the next section, the same domain is used to illustrate the potential for reuse and adaptation of pre-existing requirements definitions by comparison with the Writers' Aids work on grammar checkers (Appendix D).)

C.5.2 Overview

Requirements analysis for evaluation aims first at providing a description of the problem domain requirements. This description must support the subsequent development of detailed reportable attributes for systems under test. The requirements part proper should consist of

- a set of **tasks** described in as much detail as possible at the problem domain level;
- a set of **scenarios** or **setups**, describing the various contexts in which the tasks should be considered and the relevant variables;
- for each task and subtask, where relevant, quality requirement definitions for functional and non-functional requirements.

(The tasks and setups identified can feed directly into the test design phase of evaluation, discussed in section 2.6.)

A requirements analysis procedure for LE evaluation aims to provide guidelines for the construction of these outputs and the final output of reportable attributes. Such guidelines will never amount to a deterministic procedure. Instead, they should form a framework which can support the activity of requirements analysis by making available

- a structured set of knowledge acquisition steps and methods that can help transform generic or pre-existing task, setup and requirements into those required for the current evaluation;
- libraries of task, setup and requirements elements that have been used for other evaluations or which are in some sense 'generic' or general purpose.

An important part of the usefulness of generic or pre-existing requirements elements is the fact that, having been used in previous evaluations, they are associated with elements of the library of available test types.

The basic steps are:

[4]See Grainger (1993) for an account of the need to analyse both learner and native English for EFL applications, for example.

- Establish problem domain requirements (requirements at the outermost scope for this evaluation)
  - Identify top level tasks at the domain level.
  - Identify relevant setups.
  - Identify stake-holders and information sources for all relevant setups.
  - Identify situational and environmental variables which constitute 'user profiles'.
  - Identify functional requirements as specifications of top-level task input-output pairs, and non-functional requirements as constraints on properties of top-level results.
- Transform domain-level tasks and associated setups and requirements into reportable attributes which are measurable; this can be thought of as 'redescription'.

As previously mentioned, notions of sequence are often misleading in this sort of development activity, and the actual processes by which these results are arrived at are more likely to follow an iterative procedure, such as formalised in Boehm's (Boehm, 1988) spiral software development process model. This is because the process is both top-down (driven by problem requirements) and bottom-up (driven by the need to bottom out at measureable primitives).

C.5.3 Problem domain requirements

The first stage addresses the highest level (or widest scope) of the requirements analysis. It is in terms of this level that other levels of analysis, down to the reportable attributes which actually express measurements of systems, must be validated.

The basic tasks the system is required to address are functional requirements, which at the top level of description should relate as closely as possible to valid and measurable user requirements. For spelling checkers, this is relatively straightforward, since the main functional requirements can be expressed in terms of the difference between the spelling errors in a document before checking and those after.[5]

The top level task description is usefully supported by constructing a **process model** of the basic situation of use (or **setup**). Such a model represents the data flows and the rôles of data-transforming processes in the setup; Figure C.5.3 is an example for spelling checkers at this top level of analysis.

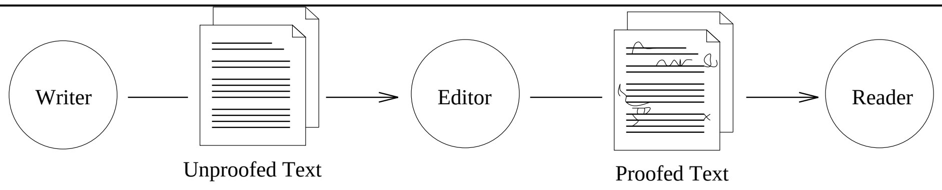


Figure C.3: S1: Top level data flows and agents in spell checking task.

Each data type and process identified in this model will be available to be decomposed and analysed further in the course of the procedure. The process can be paraphrased as something like

> (S1) The **unproofed text** produced by the **writer** is revised by the **editor** to give **proofed text** suitable for the **reader**.

[5]Note that this takes the simplest possible view of the purpose of spell-checking; a more detailed account would have to consider the rather different overall tasks involved where proposed corrections in a document have to be negotiated with the writer, or where part of the purpose is to facilitate learning by the writer, purposes which place extra contraints on the output of checking.

An immediate decomposition of this statement can be made, to bring it closer to direct description of the functionality we seek: we can say that

> (S2) The **editor** corrects the **spelling errors** in the **unproofed text** produced by the writer to give **proofed text** suitable for the **reader**.

This is based on a model of **unproofed text** which sees it essentially as a set of **spelling errors**. A spelling error is minimally a *pair* consisting of the text actually present and the intended text.[6]

Note that this level of analysis presupposes nothing about the way the checking is to be accomplished. At this level of abstraction, we can define some quality requirements on the task at the domain level. These will form the basis of more detailed requirements at the reportable attribute level.

Functional requirements can often be defined in terms of classic recall and precision measurements: does the system do all and only what it should? If we take the first task description, (S1) above, the quality requirement is just that the process in the editor rôle must transform the unproofed text to proofed text that is suitable for the reader — a quality requirement expressed in terms of overall text quality. Clearly this has to be decomposed before it says anything about spelling, as in (S2) above. However, for any such decomposition, we have to think about its validity. In a case like this one, there may not be a problem; we are likely to assume that spelling quality ought to be independent of the rest of text quality. However, keeping in mind the assumptions made at each decomposition is useful. For instance, it might be that although a spelling checker improves spelling quality by its suggestions, the change to the proofing activity that it promotes is such that overall text quality is reduced because careful hand-proofing is no longer performed. It may not be possible to validate all such decompositions, but knowing that there is a gap prevents unwarranted assumptions, and may suggest areas where warnings or more work may be needed. The introduction of a software system inevitably changes the tasks that preceded it, so although analysis of tasks before the introduction of a computer system is very valuable, it cannot be taken as the final word on the overall task.

To return to the more detailed task description and associated quality requirements... For error checking systems, classic recall and precision requirements are based on a comparison of the count of errors in the before and after texts. (The editor rôle in this process can be thought of either as a human editor in a situation before introduction of any computational tool, or as the combined role of the checking phase carried out by human and software.)

Non-functional requirements at this level might include factors like the volume of text to be checked, time constraints, and so on.

More detailed functional and non-functional requirements are to be found at the next level of analysis.

The next step is to construct a set of relevant setups, identifying situational and environmental variables that affect the requirements on the task under consideration. This includes the gathering of possibly disjoint sets of requirements from different sources, as the Consumer Report Paradigm allows. Setup identification is an outward-looking activity, based on finding real-world situations in which the product under test might be used.

Questions that are relevant for the analysis of the setup include the definition of the upstream and downstream path of information to and from the document types that form the scope of the top-level task evaluation. For example, if the text is sent to an Optical Character Recognition device after being written, this should be noted, and a new rôle node inserted into the process diagram, since this will affect the kind of spelling errors that appear in the input document. Similarly, if the downstream path of the document is not immediately a human reader, but some sort of automatic

[6] This sort of decomposition, identifying new concepts and connecting them to previous ones by relations such as **part-of**, corresponds to the domain layer of the expertise model in KADS.

process like a parser or a grammar checker, this will affect the definitions of the concept spelling error that we need to develop.

Nodes in the process model are used to structure the process of identifying variables that are relevant to task performance, and thus facilitate the modularisation of requirements. For instance, the errors present in the text before proofing are likely to be affected independently by variables associated with the writer rôle (e.g., first language and language of the text), and variables associated with the OCR rôle. Other relevant elements of the setup include the computational and organisational context.

These variables are parameterisations of the rôles in the process model. Just as different processes in the setup can constitute independent sources of variation, so can different functions within a rôle. For instance, it might be that rates of accidental typing mistakes can be treated as independent of errors of intention such as language transfer errors in second language users, though both are associated with the writer rôle.

The validity of subsequent evaluation processes depends on the validity of the methods used to analyse requirements. Sources of representative texts need to be found to characterise the before and after texts in various setups; reliable experts need to be identified to analyse them; or reliable and applicable prior research must be found that characterises these two document types. The development of well-documented corpora of representative and realistic text for a wide range of requirements is necessary.

C.5.4 Transformation to reportable attributes

After this top level definition of quality, we need to turn to a consideration of the systems we are interested in evaluating. Figure C.5.4 shows the place of the systems under consideration in the new task model:

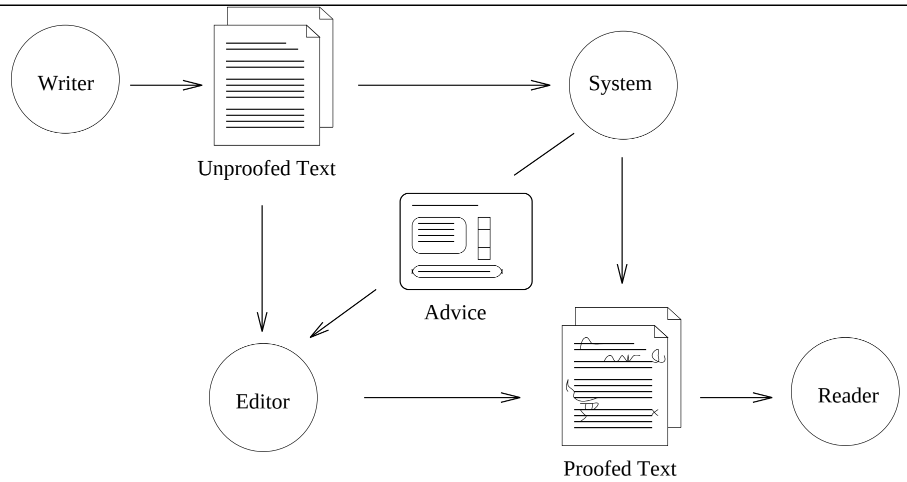


Figure C.4: S3: Introduction of computational system.

At this level of analysis, another set of quality questions becomes relevant. The rôle of the human editor, and the relations between the advice from the system and that editor, becomes available for analysis. New quality requirements can be defined in terms of the new concepts introduced. Further knowledge acquisition is required to determine the possible variable elements in different types of human editor in terms of what kind of advice is useful. At this level of analysis too, all sorts of non-functional quality characteristics of the system become relevant, from usability to compatibility with existing software environments.

This part of the modelling corresponds to the co-operation model in KADS, specifying the interactions required between system and user. Here, KA methods of the sort discussed in section C.4 are necessary for finding out what particular kinds of editor (as end-user) need in the way of advice. This may need a teacher rather than a naive user, but it may need experiments.

Up till now, the requirements analysis has all been top-down. To decompose the basic recall and precision functionality requirement into useful sub-attributes, we need to take a partly bottom-up approach based on the categories that are relevant to system performance. This must be based on some prior experience of the kinds of system under consideration, and hence will be liable to improve from repeated open-ended evaluation. For instance, it is only because we know something about the operation and limitations of spelling checkers that we might have a separate sub-attribute for their coverage of multi-word elements like *ad hoc*.

For each task, we need to enumerate and classify the ways in which different systems fulfil them.

For instance, a problem level requirement on spelling checkers for some users might be that customisations should be readily sharable between end-users. A reportable attribute called something like 'sharability of user dictionaries' might have a nominal value for each of the ways that existing or envisagable systems satisfy the requirement, including values corresponding to various kinds of failure to satisfy the requirement at a given level.

For instance, in the incorporation of the results of checking back into the text, there are a number of options, including automatic incorporation, user incorporation, version control, access.... The idea of a spelling error can be split up according to things like the kind of text function it occurs in, such as closed class vocabulary or productive vocabulary (names falling somewhere in between) since different methods of error detection work for each. Specialist vocabulary — how likely is it that this will make a difference. Development of these categories can be done to some extent *a priori*, on the basis of models of the structure of text and so on, and to some extent must be done empirically. (Cf TEMAA)

The problem of knowing what to consider testing as a factor can be alleviated by two means: the most fundamental (which should never be ignored) is the open-ended evaluation by people sensitive to the domain requirements and high level quality considerations, but the second, more related to what we can systematise in a framework, is related to the reuse and adaptation of factors that have proved useful for similar evaluations before. That is the issue considered next.

C.6 Generic requirements models: adaptation and reuse

One of the major benefits sought from the developing EAGLES evaluation framework is that of reusability. Reusable evaluation resources not only save effort in the invention of the wheel, but promote comparability and standardisation by the use of similar methods in different evaluations. The Parameterisable Test Bed (PTB) sketched in section 2.2 of the main report foresees **libraries** of attributes and associated measures and methods being built up, together with guidelines on how to put them together to make up a new evaluation. From the requirements side, too, such libraries can be envisaged. This section sketches, in the context of the application areas covered in the EAGLES work so far (grammar checkers, spelling checkers, and translators' aids), some aspects of reusability that should be relevant to the requirements analysis process, and could be included in an envisaged 'PTB+'.

We will look at aspects of the requirements analysis process covered in the last section, and for each, consider what use can be made of commonalities, and what constraints we need to place on representations in an envisaged PTB+ to make this possible.

In the previous section, the first step was to construct a description of the task at a level that provides a reasonably valid representation of user requirements, separate from system considerations. It was suggested that it is useful to start from a simple process model of the situation, which represents the data flows and the rôles of data transforming processes in the setup.

For tasks which are essentially document transformation filters, this is relatively straightforward, since the state of the document before the filter is a given (dependent on the setup), and the required state of the document after can usually be determined by analogy with human processes. The system under test can be evaluated, on this highest level, in terms of comparisons between two document types, the input and output to a process very much like that illustrated in Figure C.5.3. All proofing-reading, including spelling checks and grammar checks, can be described at this level of abstraction by the same task model. Even translation can be accommodated to the same model. The task description might be given as

(F1) The **filter** transforms the **text** produced by the **writer** into the **text** required by the **reader**.

The commonality can be maintained at the level of the next process model, which includes a human and software co-operation in the task, and hence allows evaluation of the quality of advice. The task model at this level has the basic structure of a particular sub-type of text transformation systems, namely interactive or computer-assisted text transformation systems, and once again this is more or less applicable to both writers' aids and translators' aids.

Figure C.6 illustrates the relationships.

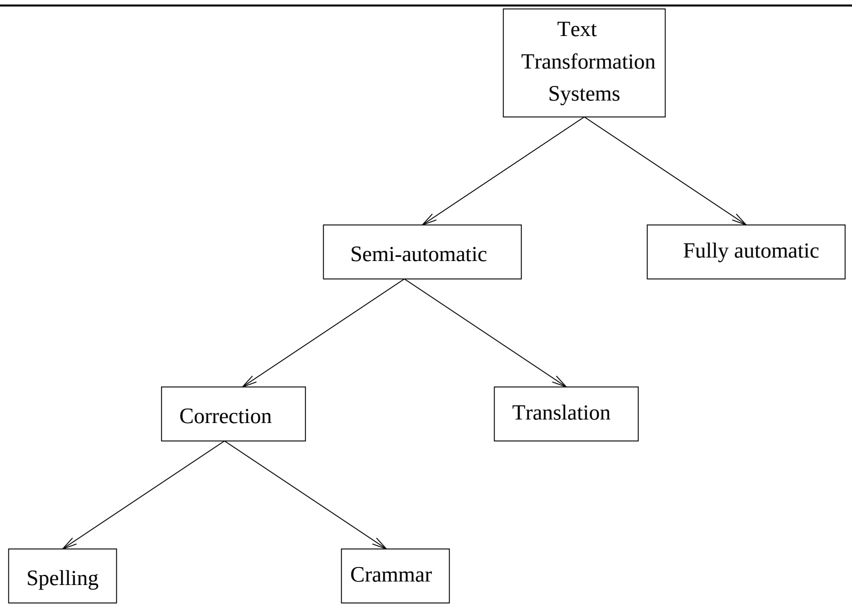


Figure C.5: Part of an Abstraction Hierarchy for Text Transformation Systems

The benefit of identifying the commonality is that once a common task model at a particular level of abstraction has been developed, requirements for particular quality characteristics that are relevant at that level can be associated with that model, and are then available as prompts to suggest the investigation of their relevance for new evaluations with setups which appear to match the existing task model. Identifying hierarchies of task models and associated requirements allows potentially relevant requirements to be inherited from higher levels. For instance, for all nodes underneath **Semi-automatic** in Figure C.6, it will be relevant to consider quality requirements to do with the suitability of the information presented by the system for the particular end-user. Some simple quality requirements can be fully expressed at this level, for instance, the requirement that the language of the information presented by the system be one in which the end-user is adequately profficient. This is independent of whether the information is about suggested corrections or suggested translations.

Task descriptions, such as (S1), (S2) and (F1), consist of a number of data processing rôles (writer, reader, filter, editor...) and data types (text, error...). These have parameters for any factors associated with these rôles that affect requirements. These rôles and data types can be thought of as arranged in inheritance hierarchies, so that for example all data types that are subtypes or specialisations of **text** will have a parameter **language**; this serves as the basis for a simple quality requirement that can be applied to all systems of this type, namely that the nominal language which the system deals with should be that of the text it is required to process.

The language of the end-user and of the system presentations are parameters that ought to live at the level of node **Semi-automatic**, together with the quality characteristics that depend only on them. More detailed requirements about the style or nature of the information belong lower down in the tree, where the task descriptions become more specific and requirements can be expressed in terms of suggestions for replacements, or diagnoses of errors.

New evaluations can be classified in a hierarchy like that in Figure C.6 by characterising the tree as a discrimination network based on differences in the values of parameters as well as the basic structures of the process models. If a new evaluation 'fits' down to an existing level, but doesn't fit any of the children of that level, a new child node can be constructed. The methods by which its sibling nodes were decomposed from the common parent may be used as guidelines about how to fill in the task model and requirements at the new node.

For instance, if we have evaluations for spelling checkers in a number of languages (French and English, say), we may want to add one for a new language (Danish, say). At the very top level, as it happens, the **text** input and output to the filter process has a parameter **language**; we would realise at this point that none of the available values were suitable. Then we'd work our way through the discrimination tree to the point where the difference in language makes a difference to a decomposition of a task description, somewhere below the **Spelling** node where the specific errors begin to be described at a level that is language-specific. Having submitted a new value for the language parameter, none of the relevant children at this level would match, since they would have values filled in for the specific language in which they dealt. If, however, there are specific knowledge acquisition methods associated with the decomposition of the idea of spelling errors into English spelling errors and French spelling errors, such as corpus gathering and analysis methods, or even pointers to types of literature that might contain relevant information, these 'intensional' descriptions can be reused for the new language.

It is not yet clear what the best way of formalising the structure of task descriptions and requirements elements is. The foregoing discussion is clearly illustrative, suggestive even, rather than precise. There appear to be a number of representations or approaches that might be suitable; systemic networks, models of inheritance in the lexicon, straightforward object-oriented methods and classic KR techniques should be considered for further elaboration of the sketches here.

C.7 Conclusions

It should be clear from this document that much work remains to be done in the area of requirements analysis and user profiling for LE evaluation. A few points can be pulled out for particular attention.

- More work needs to be done on requirements analysis guidelines for LE evaluation, both in the design and accumulation of libraries of task, setup, and requirements elements, and in the development of a set of structured knowledge acquisition steps for the procedure.
- Insights, tools, and procedures from many of the research areas in the requirements engineering community (such as traceability of requirements, viewpoint analysis, and validation of requirements) could be made applicable to LE evaluation with a bit of work.
- Requirements elements could be reused in an extended Parameterisable Test Bed, in the style of the Generic Task Libraries of the KADS modelling system, if they can be suitably represented. Aspects of the requirements analysis process that could be supported by such a tool

would include the development of setup and scenario descriptions from pre-existing elements, and the recording of justifications/validation information for requirements transformations. Recording the process of requirements analysis serves to legitimate an evaluation in the same way as recording the methods used to measure system performance.

- Knowledge Acquisition techniques used for Knowledge Based Systems could aid the systematic development of specifications for LE systems. Automated toolkits for KA could be adapted to support requirements gathering for evaluation.

- Specifically linguistic modelling based on analysis of realistic corpora needs to be integrated into any such scheme. Characterisation of corpora in such a way as to support the definition of modular, hierarchical requirements elements is an interesting challenge.

Appendix D

Evaluation of Writers' Aids

D.1 Introduction

In this appendix a first attempt is made to investigate how the EAGLES framework can be instantiated or linked to the actual evaluation of writers' aids. In Appendix I the framework is applied in an evaluation of one specific grammar checker.

D.1.1 Reviews

This section describes three different reviews of grammar checkers. The three reviews serve as a starting point for the development of the quality characteristics specific to grammar checkers which are presented later in section D.3.

D.1.1.1 Notes from "Intelligent text processing: A survey of the available products" by Shona Douglas

This report (Douglas, 1990) describes the capabilities of some of the most common automated writing aids, primarily from a linguistic perspective, but also dealing with user-friendliness in terms of the user interface and the degree of customisability offered by them. The evaluation was carried out in 1990 on a set of six then commercially available grammar and style checking programs (Correct Grammar, Grammatik, MacProof, RightWriter, Sensible Grammar, and StyleWriter). The evaluation had two aims: firstly, assessing the adequacy of the systems, to ascertain the commercial state of the art, and secondly, assessing the kinds of technology used and how this related to the adequacy performances displayed. The second aim was intended to provide input to the design and development of the automated writing aid *The Editor's Assistant*, at the University of Edinburgh.

D.1.1.1.1 Summary of method The two-fold aims of the investigation are mirrored in two linked taxonomies. The first taxonomy, the *user centred taxonomy*, provides a list of error types, intended to provide a system of categories by which the errors detected by systems would be classified so as to correspond to the user's conceptual model of the domain of activity. The second taxonomy, the *system centred taxonomy*, aims to present a characterisation of possible underlying mechanisms used by different systems, on a number of dimensions. The two taxonomies are linked by the fact that detecting and correcting particular errors (or examples of errors) was deemed to require a certain level of linguistic technology. The aim of the evaluation was to use the user centred taxonomy to position the system under test within the system centred taxonomy, i.e., to determine, from the results of a set of tests based on the user centred taxonomy, what level of technology the system under test utilised, and thus its potential range of operation. This is a sort of general diagnostic evaluation, not unrelated to the idea of reverse engineering.

D.1.1.1.2 The system centred taxonomy The dimensions of the system centred taxonomy were:

1. Representing the text;
2. Recognising patterns in the text;
3. Responding to errorful text;
4. Manipulating the text representation; and
5. Customising the rules.

The first three dimensions are self-explanatory and should be clear to the reader; dimension (4) relates primarily to interface characteristics — how the error reports are delivered and how easy editing is. Dimension (5) refers to how much freedom to change the operation of the system is in the hands of the user, for example switching on and off rules or writing new ones. For dimensions (4) and (5), the systems were classified more or less by inspection or consulting the user literature.

The numeric results assigned to dimensions (1) and (2) were intended to reflect the overall potential capability of the system's underlying technology to detect and respond to grammatical errors. The division into how the text is apparently represented and what kind of patterns the error rules can find in the text was intended to clarify the level of linguistic sophistication of the underlying methods used by the various systems. These dimensions, and to some extent the third dimension (which reflects whether responses from the system seem to be all canned text or whether they have variable pattern elements), form a group of properly linguistic objects of evaluation.

D.1.1.1.3 The user centred taxonomy: A high-level specification of functionality The user centred taxonomy is divided into four top-level sections, *grammar*, *punctuation*, *style* and *usage*, of which only the first is further considered here. Grammar errors are subdivided into agreement and inflection errors, errors in the use of relative and reflexive pronouns, errors in comparatives and correlatives, problems with negation, and numerous heterogeneous categories. Punctuation errors are considered according to the following classification of punctuation functions: parenthetic marks, sentence terminators, sentence punctuators, spacing and hyphenation. Moreover, problems concerning how these can appear in combination are covered.

Concerning style, the aim is to write clearly, directly and in an easily comprehensive English. Among the style errors, the use of the passive voice, nominalisations and the use of over-long or complex sentences are mentioned.

Typical usage errors are capitalisation and double word errors. Some other errors are due to confusion such as phonetic similarities. In other cases, they are concerned with the appropriate use of language in terms of genre or dialects, such as British English *vs.* American English and formal *vs.* informal style. In addition, since not all documents will have the same requirements, *house style* is mentioned as a way of imposing consistency over a wide range of characteristics of text.

Associated with each error type are examples and *requirements*. The latter are intended to reflect what underlying linguistic sophistication a system would need to successfully detect the error in question and hence they provide a link from the user centred taxonomy to the system centred taxonomy.

D.1.1.1.4 The test suite The values assigned on the dimensions of assessment derive from the results of applying a test suite based on the user centred taxonomy. The test suite was composed of an extension of the examples in the user centred taxonomy. Each error type has its corresponding set of examples in the test suite. Test examples are constructed in both positive and negative modes, that is, where an error exists in the construction and where one does not.

D.1.1.1.5 Scoring the tests The set of possible outcomes for each example reflected the dual aim of testing for positive and negative success. For an example marked with an asterisk, to denote the fact that it contains an error, there are five possible outcomes, described in table D.1.1.1.5.

Table D.1: *Outcomes for examples marked with an asterisk*

Outcome	**Explanation**
$\surd$	response identifies error clearly
0	system fails to respond to error
$\frac{1}{2}$	the error *could* be diagnosed from the response, especially if it is an error of execution (i.e., the user only needs his attention drawn to it to recognise it)
?	there is a response, but sufficiently indirect to make error diagnosis difficult
!	response completely unrelated to error (i.e., a false positive showing up in part of the suite not specifically designed to trap it)

For an example not marked with an asterisk, the outcomes shown in table D.1.1.1.5 are possible.

Table D.2: *Outcomes for examples not marked with an asterisk*

Outcome	**Explanation**
␣	correctly ignores the correct text
$\times$	falls into false positive trap
!	particularly awful false positive

Apart from the use of the exclamation mark, which was intended to draw attention to some particularly nasty example of false positive, there was no explicit weighting system.

D.1.1.1.6 Progress evaluation for *The Editor's Assistant* The system-based aspect of the taxonomy of errors that was developed for adequacy evaluation was the basis for the progress evaluation scheme for *The Editor's Assistant*. Error types were grouped into four categories, viewed from a computational linguistic point of view:

1. **Constraint violation errors:** These involve what, in most contemporary syntactic theories, are best viewed as the violation of constraints on feature values. All errors in agreement fall into this category, for example *There are violence in the school* for *There is violence in the school.*
2. **Lexical confusion:** These involve the confusion of one lexical item with another. Specifically included in this category are cases where a word containing an apostrophe is confused with a similar word that does not, or *vice versa*. In practice, cases are limited to where the confusion results in an ungrammatical sentence; that is, where the confused words are of different syntactic classes. For example: confusion of *its* and *it's*; confusion of *there*, *their* and *they're*; confusion of possessive *'s* and plural *s*.
3. **Syntactic awkwardness:** Included in this category are cases where the problem is either stylistic or likely to cause processing problems for the reader. These errors are not syntactically incorrect, but are constructions which, if overused, may result in poor writing,

and as such are often included in style-checker hit-lists. Thus, multiple embedding constructions, potentially ambiguous syntactic structures and garden path sentences are included in this category. A typical example of a garden path sentence is *The horse raced by the barn fell*[1]. These problems are detectable by simple counting or recognition of syntactic forms.

4. **Missing or extra elements:** These are cases where elements (either words or punctuation symbols) are omitted or mistakenly included in a text. For example: unpaired delimiters; missing delimiters; missing list separators; double syntactic function, etc.

These error types are associated with general error detection/correction mechanisms in *The Editor's Assistant*. The third type (associated with syntactic awkwardness) was similar to pattern-recognition algorithms, since it merely recognised patterns in the analysed text. The other three types were based on different *error hypothesis* schemes and relied on an extra processing loop to test whether performing the suggested correction on the hypothesised error actually led to a better sentence. Work focused on developing implementations of these general mechanisms.

D.1.1.2 *PC Magazine Laboratories' performance tests*

In May 1993, *PC Magazine* published a review of writers' tools based on their own performance tests (Rahmstorf and Rabinovitz, 1993). In this section we will briefly describe the methodology adopted by *PC Magazine* Laboratories in performing grammar checker tests.

The systems reviewed were Correct Grammar for Windows, version 2.0 and Correct Grammar for DOS, version 4.0, Grammatik 5 for DOS and Windows, Rightwriter, version 6, and CorrecText.

The test material comprised examples of typical punctuation and capitalisation errors and 75 typical grammar and style errors. Each error occurred in two different sentences to avoid the potential problem that a single example would be unusually easy or difficult for a particular program to detect. In many cases a grammar program could detect only one of the examples of a problem.

The evaluation of the output for each sentence involved giving one of four possible responses and assigning an appropriate score, as shown in table D.1.1.2.

Table D.3: *PC Magazine Laboratories — responses and scores*

Response	**Score**	**Comments**
Hit	1	The program flags an error.
Miss	0	The program does not detect an error.
Advice	0–4	The program gives one or more good suggestions about how to correct an error. Each sentence with this response was given a score from 0 to 4 according to the quality of suggestions given by the program (a score of 4 means that the error was flagged and all elements contributing to the problem were detected).
False flag	<0	The program detects a non-existent error or recommends a correction that introduces an error. Negative points were given for flagging nonexistent errors or providing advice that introduced new errors.

An additional score used is *Hits-to-False-Flags Ratio*, being the relative number of hits and false alarms. The higher the ratio, the less likely that the program will report nonexistent errors or offer bad advice.

[1] In this sentence *raced* has to be reinterpreted as a pastparticiple form of the verb instead of a past tense form.

Grammar checkers were also evaluated for their spell-checking abilities. For this purpose, the test material contained 100 misspelled words of ten different categories. 1 point was given for detecting a misspelled word and suggesting the correct alternative.

D.1.1.3 Chandler's review of grammar checkers

Chandler (1989) reviewed a number of grammar checkers for *Electric Word*. Chandler, an English teacher at the University of Hawaii, was interested in assessing the usefulness of grammar checkers for his students, who included a large number of Asian second language students. The systems he tested were Grammatik III, Correct Grammar and Critique. He assessed the systems not only for their linguistic accuracy in detecting errors (grammatical and usage errors) but also for their user-friendliness. Under the second category he considered factors such as the helpfulness of the tutorial (one that presented examples was more helpful than one that just stated rules) and the type of editor. Chandler also considered it helpful if parse trees were presented to the user. Additional factors that he took into account were how easy it was to add words to a system's dictionary and the price of the system.

Chandler's assessment of the systems according to the dimensions outlined above appear to be entirely subjective, except in the assessment of error detection, where he used a very small test suite. He ran the test suite twice through each system. The phenomena tested were intended to contain typical errors made by Asian second language students. Errors included grammatical errors (including agreement, comma splices, and subordination and modifying phrases) and usage errors (e.g. *a* for *an* and *childrens* for *children's*).

D.1.1.4 Conclusions

The evaluations described in the above reviews differ vastly in scope and rigour of approach. Nevertheless, there is a fair amount of agreement on which characteristics are important when assessing the performance of a grammar checker. These performance characteristics or functionality characteristics include:

- Detection of error types (grammar, punctuation, style, spelling);
- Specification of errors and quality of advice;
- Avoidance of false flagging.

In addition to these performance characteristics, a number of other characteristics were reported on that are considered important in assessing grammar checkers, including:

- Customisability;
- Ease of use;
- Compatibility.

D.2 Selection of grammar checkers to be used as test beds

The group has used one grammar checker for English and one for French as test beds for the development of a methodology for testing grammar checkers. Since we are not concerned with the evaluation of the particular software products, we will not mention the actual names of the grammar checkers used for our purpose. Therefore we will refer to the English product as *E1*, whereas the French one will be identified as *F1*. Both grammar checkers include spelling facilities.

D.2.1 Reasons for the selection

Our selection of grammar checkers is based on:

- Availability;
- Price;
- Language;
- Second language facility;
- Compatibility with word processors; and
- Spelling checker included

D.2.2 Focus of attention

Given the limited amount of manpower available, the group has concentrated on the *grammar* checking functions of such products.

Consequently, certain areas are deliberately **not** considered by the present study. These comprise:

- Punctuation;
- Document structure and capital letters;
- Collocations and idiomatic expressions;
- Wrong lexical choices;
- American English; and
- Style and usage.

D.3 Quality characteristics

In the framework of evaluating grammar checkers we suggest the following specialisations of the qualitative characteristics, based on ISO 9126:

Functionality:

- the system should detect errors and specify what they are, thereby enabling the user to correct them;
- the system should not specify non-errors;
- the system should be able to handle combinations of errors to some extent.

The present study focuses on this characteristic, since this is the one that has to do with NLP.

Usability:

- the user interface should have a form that makes the system easy and efficient to use, e.g.
  - dialogue;
  - windows-based;

- text replacement;
- marked-up copy.

Reliability:

- the system should not crash or loop;
- the system should always specify the same grammar error in the same context the same way.

This characteristic will not be tested explicitly.

In addition to the characteristics proposed by ISO 9126, we suggest:

Customisability:

- the system should allow the user a certain control over the functionality.

Customisability could be said to be part of functionality, but in our opinion it is a separate characteristic.

Below, we concentrate on functionality, and the attributes, measures and methods attached to that particular quality characteristic. **Efficiency**, **maintainability** and **portability** we find are of importance to software developers and maybe certain purchasers of systems, but not relevant to the NL focus of the present work. First, we consider the task in more detail.

D.4 Task model

A model of the task setup, including the system class being tested, will be used in elaborating a detailed set of requirements from the basic quality characteristics we have identified. These requirements will be used in the derivation of reportable attributes and of measures and methods for arriving at values for those attributes.

For grammar checkers, the relevant task is the *revision phase* of the writing process. The agent and output types we recognise in this task are presented in the following task outline:

> A *non-proofed text* prepared by the *writer* is revised by the *end-user* using *advice* produced by the *system* to produce a *proofed text.*

In this model, the final output *proofed text* is an ideal result, which we can use to measure actual functionality.

In this particular mass-market system type, the writer and the end-user are usually the same person. However, this is not necessarily the case and the different roles are distinguished because different characteristics of the person are potentially relevant to each. The task outline given here and many of the elaborations of the various components we give below are general to all revision-stage setups, including spelling checking [2].

Relevant parts of the task model will include linked sets of descriptions:

- A set of *input* models that corresponds to the set of *subjects* of the various actions in the task outline: the writer, the end-user, and the system;

[2] There are even characteristics that can be stretched to all text transformation setups, such as translation, but that will not be pursued further here.

- A set of *result* or *output* models that corresponds to the *objects* of the actions in the task outline: the unproofed text, the advice and the proofed text. A taxonomy of *errors* is a derived result produced by comparison of the unproofed text and the target proofed text.

Each of these will be treated in turn, in the order of their place in the process, and then the relationships between them examined.

D.4.1 Model of writer types

This covers writer attributes that are relevant to the writing task, to the unproofed text that is produced and the errors that are likely to be in it (where errors are defined as the difference between the unproofed text and the target proofed text). Revelant characteristics include:

- Proficiency in target proofed text language;
- Professional training;
- Intentional error sources (dialect form, genuine misconceptions, etc.);
- Medium-related error sources (cut and paste, OCR, typing, etc.);
- Performance related error sources (attentional slips, etc.).

We cannot assume that the writer has exactly the goals of the end-user (or reviser) in terms of the target proofed text. For example, a text originating with an American writer intended for a USA audience might be revised with British English as a target. Thus, the definition of relevant idiosyncratic intentional error sources is dependent on what the actual proofed text target is; there may be many other idiosyncratic error sources that are not relevant to the actual task target.

D.4.2 Taxonomy of systems

Under one view, a taxonomy of systems is the primary output, or result model, and should correspond to the set of reportable attributes that form the columns of the Consumer Report. However, at this stage we think of it as an input part of the model and consider only the following simple characteristics that can be established without testing, and which determine relationships with other input types such as end-user language proficiencies:

- Proofed text target language
- Language of advice

D.4.3 Model of end-user types

The usefulness of system advice of different types and qualities depends on characteristics of the end-user, such as

- Linguistic education (for possible use of terminology in advice);
- Language proficiency in the unproofed text language (for understanding the intention of the writer or the source of slips);
- Language proficiency in the proofed text language (for recognising or generating correct replacements);

- Language proficiency in the language of the advice (for recognising correct suggestions where these are discursive rather than replacement suggestions, and not necessarily couched in the language of the target proofed text).

We are using the term *language* here in rather a comprehensive sense: when we talk about an end-user's proficiency in the language of the advice, for example, we mean not only whether the language (French, for example) is one understood by the end-user, but also whether the type of explanations offered are comprehensible in terms of terminology and structure.

D.4.4 Model of unproofed text

The nature of the unproofed text depends on the writer's characteristics and those of the task the writer is engaged in. In theory, a model of unproofed text would be a detailed description of the language the writer produces, both 'correct' and 'incorrect' parts without distinction. In practice, for the detailed matter we substitute the derived taxonomy of errors, which together with the proofed text taxonomy should generate the unproofed text taxonomy: bear in mind that a model is just something that behaves in the same way as the thing modelled, under certain circumstances. The more general characteristics of the unproofed text are :

- The language in which it is written (e.g., French or English);
- The genre and/or style of the text are also relevant. In principle, we might include here whether a text is supposed to be written in some specific *controlled language*.

D.4.5 Model of proofed text

The proofed text type is in some sense primary: it is in theory a complete linguistic description of the target language in question. The actual instantiation of this in some relatively formal representation will be a major task, where it proves necessary. In practice, the detailed and explicit part of this description is only necessary for the purpose of making tests: it may be possible to substitute a method of generating test examples that is adequate, as long as the distinctions stated in this taxonomy can be shown to be instantiated. Thus, this part of the model, in our portable, self-contained definition of the task and attributes associated with it, may be mostly implicit. General characteristics that should be given to guide the later processes include:

- The target language of the text (for grammar checkers, this must always be the same as the language of the unproofed text; we are dealing with text transformation but not to the extent of translation);
- The target style of the text (this might include ideas about the intended audience).

D.4.6 A Taxonomy of grammar errors

The taxonomy of errors can be thought of as being derived from a combination of the proofed text model and the errors likely for particular writer types. Alternatively, it can be thought of as derived from analysis of texts before and after the notional process of copy-editing, that is, comparison of the unproofed text model and the proofed text model. In both cases, we are interested in errors that actually occur. The purpose of the taxonomy is:

- To permit the classifying of errors actually encountered in text, to establish classes of frequency and importance, both with respect to writer types and possibly text types, and in aggregate terms;

- To form the basis of reliable test methods for the performance of systems;
- To enable the results of testing to be mapped on to the customer-driven presentation of results — the *reportable attributes*.

Given these purposes, we must establish a principle to use in classifying grammar errors. The two derivations of the taxonomy we have offered (proofed text plus writer error sources, or proofed text compared with unproofed text) come from different directions, and must be brought together to form a useful classification.

If we consider the approach from the proofed text and the sources of writer errors, we could classify errors in terms of their source in the writing process. A few obvious types in this classification, as mentioned in the writer model, would be: slips of medium (typing errors, OCR errors, cut and paste slips...); dialect differences between the writer's language and some standard language; second language errors; concentration lapses resulting in 'derailed' sentences; and other performance errors. Such a taxonomy has the advantage that if we have a proper writer model, we cover all errors that result in ungrammatical text, and it may fit the writer's and end-user's categories of thought and thus permit easy mapping on to customer-reportable attributes, which is an important purpose of the taxonomy. However, our writer model would then have to be a detailed psycholinguistic model of language competence and performance, and this seems rather a tall order. In practice, the source of our writer model is likely to be an analysis of proofed and unproofed texts, that is, working back from the second type of derivation of the taxonomy of errors.

Comparing proofed and unproofed texts as a way of arriving at errors has the practical advantage that it is definitely true — as far as it goes. We can relate this comparison to the traditional copy-editor's marks which simply give editor operations such as moving, deleting, adding, transposing and so on. However, this does not contain much information about the linguistic causes or categories underlying errors and thus gives us no useful categories upon which to base frequency analysis or reporting attributes, or test generation. Even for a single user, a given error example could have multiple possible causes in such a taxonomy; only the writer could possibly tell, and possibly not even them. In human copy-edited text, the semantic information is supplied by the reader (the end-user in our model), and in creating writer models and error taxonomies from proofed and unproofed text pairs the researcher must attempt to produce a classification that is informed not only by actual text occurrences but by some idea of source.

Aspects of both approaches are combined in the design of our taxonomy. The taxonomy is arranged in terms of the writer sources of errors, to give the semantics that will support the end-user task, each mapped on to a set of transformations of the model of proofed text at the analytic level appropriate to the error type. Some examples:

- OCR errors will be represented by typical letter or bigram substitutions in the end-product (literal output) of the proofed language model;
- Second language errors such as wrong prepositions in prepositional phrases will be represented as lexeme substitutions or possibly as subcategorisation errors if they are closely linked to verb frames;
- Concentration lapse errors or planning fault errors such as failed agreement between noun phrase determiner and head will be represented as the breaking of a constraint in the grammar of noun phrases.

D.4.7 Taxonomy of advice types

This comes from a combination of the system model and the end-user model. We need a taxonomy of the responses given by systems that not only covers what will come out of grammar checkers, but that also covers what actual advice various end-user classes would need to allow them to correct an error — what ought to come out of grammar checkers, as it were.

The structure of a response is an *ordered set of suggestions*, each of which has a given *advice type*, and the instantiation of that type has a certain *advice quality*. Characteristics of advice are:

The number of suggestions and their order: This is related to precision/recall questions concerning how easy it is for the end-user to choose among the suggestions.

Type of diagnosis support: By this we mean any material that accompanies a suggestion designed to enable the end-user to decide whether to accept that suggestion or not. Where there are multiple suggestions the task is choosing among the set of suggestions plus the option of ignoring them all. The kinds of value we identify are:

- Flagging only;
- Flagging only with suggestions — this is typically the case for spelling checkers — the suggestions themselves help with diagnosis;
- Name of error type;
- Set of examples of the hypothesised error;
- Text describing hypothesised error.

Type of suggestions: We identify the following main kinds of value for this characteristic:

- Offering an explicit replacement for an identified stretch of text;
- Offering a piece of instructional text that specifies how the end-user should transform the flagged text into the replacement;
- Offering a piece of diagnostic text that supports the user in recognising whether there is an error of the type suggested as well as general instructions on how to fix it;
- Drawing attention to a stretch of text in a general way.

Certain pieces of advice may contain more than one of these elements, which can confuse matters.

D.4.8 Relationships among parts of the model

We have already seen how the various parts of the task model are interdependent and can be defined and obtained in terms of various combinations of the others. This has implications for how we fill in and validate the model, and how we obtain in the detail required for those parts of the model that we need in order to proceed with our testing and reporting.

The error taxonomy and the proofed text model together are, in a sense, primary in terms of our evaluation methods. All the other taxonomies have aspects that are only fixed relative to the errors, whether inherently or as a result of the practicalities of collecting information, and the errors are only fixed relative to the proofed text model. Thus, we find the following:

- The error sources part of the writer model is likely to be built up by examining a (large) set of unproofed and proofed texts from a given class of writers, and allocating frequency measures to errors that are classified by the researcher in the pre-existing taxonomy;
- The unproofed text model (not to be confused with instances of unproofed text, which have yet to be classified and hence cannot be used for the same things) can be considered as the proofed text model (which does not vary with any of the other factors) plus the error taxonomy;

- The end-user model tells us, for a given error type, what quality of advice of a given type is necessary to correct it.

However, as we have already discussed, the error taxonomy itself is derived from examination of unproofed texts, implicit knowledge of proofed text and shrewd but unfounded ideas about error sources. Are we going round in circles? Not quite; Appendix C contains some preliminary discussion of the issue of validation for these structures, but the problem does require more work.

In our evaluation method we use these linked structures of the model in a number of different ways; different parts of this linked structure must be fully realised to support these different purposes of the model, which are similar to the purposes given for the error taxonomy:

- To support the statement of detailed requirements;
- To form the basis of reliable test methods for the performance of systems;
- To enable the results of testing to be mapped on to the customer-driven presentation of results — the reportable attributes.

D.5 Requirements

We have already provided some application-specific elaboration of the idea of functionality, the equivalent of a set of high-level requirements:

- The system should specify the grammatical errors made by the writer in such a way that the end-user is able to correct them;
- The system should not signal an error where no such error is present;
- The system should be able to handle combinations of errors to some extent.

We have also provided the outline of a task model that makes clearer the parameters along which detailed understanding of these requirements may vary. In this section, we bring the task model and the high-level requirements statement together to form a detailed statement of requirements that will form the direct basis for testing and reporting.

It should be understood that the following illustrative discussion is at an extremely preliminary stage.

D.5.1 Functionality requirement — part I

> *The system should specify the grammatical errors made by the writer in such a way that the end-user is able to correct them.*

This requirement can be restated as a property of a relation among the error taxonomy, the writer model, the end-user model and the advice model that may or may not hold for a particular system under test. To illustrate this rather complicated idea, we will consider a particular error — determiner-noun number disagreement in English noun phrases (abbreviated as EDetNNum) — and show how the other parts of the model relate to the error taxonomy. We examine each major element of this requirement in turn.

D.5.1.1 The grammatical errors made by the writer

For each class of writers, an EDetNNum error has a certain frequency or significance level, derived from empirical analysis of texts. This is one way of instantiating the various sources of errors associated with particular writer classes in the writer model. So we can think of some set of instantiated writer models (e.g. one for L1 English writers, one for L2 English–L1 French, one for L2 English–L1 Danish...) with their error sources expressed in terms of the classes in the error taxonomy:

```
WriterClass1:
Language proficiencies: English 10 ...
Error sources: intentional: EDetNNum 1; ANOtherError 1...
               performance: EDetNNum 3; ANOtherError 2...
               medium: EDetNNum 5; ANOtherError 5...
...

WriterClass2:
Language proficiencies: French 10
                        English 5
                        ...
Error sources: intentional: EDetNNum 4; ANOtherError 4...
               performance: EDetNNum 4; ANOtherError 4...
               medium: EDetNNum 5; ANOtherError 5...
```

These error frequencies can be viewed from the other direction, as contributing significance values to elements of the error taxonomy:

```
ErrorClassEDetNNum:
Frequencies: WriterClass1 9
             WriterClass2 13
             ...

ErrorClassEModNNum:
Frequencies: WriterClass1 8
             WriterClass2 13
             ...
```

We can sum up these significances, if they are evenly distributed among the relevant classes, or we can keep them separate if, as in this case, different classes of writers make different numbers of errors. This will be a consideration when we come to choose reportable attributes.

If this significance value is above a certain threshold (this is a property of the relation between writer models and error taxonomy), we can say that EDetNNum is one of *the grammatical errors made by the writer*, and methods should be devised for testing system performance on that error.

D.5.1.2 The specification made by the system

In terms of our task model, *the system specifying the grammatical errors... in some way* is a relation between systems, errors and advice. For a particular system, we could have:

```
System1:
Advice: ErrorEDetNNum: Number of suggestions: 1
                       diagnosis support: name of error type,
                                          examples
                       suggestions: explicit replacement
        ANOtherError: ...
```

```
System2:
Advice: ErrorEDetNNum: Number of suggestions: 1
                       diagnosis support: name of error type,
                                          examples
                       suggestions: general instructions
```

This is the main relation that is to be determined by testing. We test a system and fill in the advice given for types of errors, according to some method. This will be dealt with in the section D.7 on methods.

D.5.1.3 The correction of errors by the end-user

The element *In such a way that the end-user can correct the error* is a relation between errors and the advice an end-user needs to correct them. Thus, it provides a criterion or threshold value against which to test the information collected under the previous heading (*the specification made by the system*). We might have, for a number of classes of end-user:

```
End-user1:
ErrorEDetNNum: Advice: Number of suggestions: ANY
                       diagnosis support: name of error type,
                                          examples
                       suggestions: drawing attention

End-user2:
ErrorEDetNNum: Advice: Number of suggestions: 1
                       diagnosis support: name of error type,
                                          examples
                       suggestions: explicit replacement
```

With this information, we can work out that both System1 and System2 would pass the test for EDetNNum as far as End-user1 is concerned, but only System1 would do so for End-user2, because the latter requires explicit replacement suggestions to make the change correctly.

D.5.2 Functionality requirement — part II

The system should not signal an error where no such error is present.

This part of the requirement is linked to our consideration of properties of grammar checkers related to the information retrieval property of *precision*. It will not be dealt with further here, except to note that it will require us to add detail to our ideas of what signalling an error means in terms of flagging a part of unproofed text. However, there will be some mention made of precision in the discussion of reportable attributes (D.6) and of methods (D.7), because it provides information that is necessary to make sense of the part I requirements for the customer, and because it requires a different kind of testing method.

D.5.3 Functionality requirement — part III

The system should be able to handle combinations of errors to some extent.

This part of the requirements will not be dealt with further here. It should be noted that some empirical work on how frequent error combinations are, and what sort of combinations occur, would be useful.

D.5.4 Discussion

There is a number of points to note about the process that has been sketched here for the part I requirement (D.5.1). First, it is designed to provide a way of combining and comparing information from various parts of the task model in order to express the requirements in more detail. Secondly, it should be made clear that not all parts of the model need be individually present; many values can be derived from defaults, aggregate scores and so on. It is also the case that in practice we will often not be able to make all the distinctions present in the model; the model indicates what ought to be considered, but practical difficulties may interfere. The problem of ranking advice according to end-user understanding, when advice from grammar checkers comes in so many different and non-comparable forms, is an example. However it is felt that stating the requirements in this way helps to clarify the issues, and, for example, allows us to motivate differences between the evaluation methods possible for grammar and spelling checkers.

D.6 Reportable attributes of writers' aids

The task addressed in this section is the choice of *reportable attributes* that will form columns of a Consumer Report. Each of the quality characteristics mentioned in earlier sections will give rise to a number of such reportable attributes.

The attributes we seek to identify are descriptions of a system's performance in a given setup which allow the customer (the user of the evaluation) to make the decisions they require about a set of systems in a class. We have just seen how a combination of the requirements and task model can be used to give detailed requirements statements that can be based on measurement. Our task here is to decide what to consider when reporting the results of these tests to the user.

The choice of reportable attributes is driven by considering the requirements of the customer — the user of the evaluation. In general, the customer and the end-user (the user of the system) may have different requirements, and they play distinct roles in the design of the evaluation. The customer's requirements may be based on the impact of the whole end-user/system setup that is being evaluated on wider tasks in a wider corporate setup. In the case of grammar checkers (which are relatively mass-market products commonly used by non-corporate writers) the customer and the end-user may often be the same person, but in principle the end-user who actually interacts with the system is a variable in the setup the customer is evaluating. Thus, arriving at a list of reportable attributes involves an analysis of the interaction of the product and the end-user in the immediate task, and of the place of this task and its outputs in the wider task that may interest the customer. We have already outlined a model of the immediate task, which is relevant both to this choice of attributes and to the development of test methods for the functional aspects of the systems. In this section, a further model, of customer types, is in theory called for.

In the case of grammar checkers, and particularly the functionality attributes we are currently concentrating on, we feel that customer and end-user perspectives are not likely to differ significantly [3]. This is particularly true of the quality characteristic we are concentrating on in this report, functionality. Different characteristics, however, will be relevant to each role, and so we continue here as if our end-user model has no relation to our new customer model.

The customer must be analysed to find out how the functionality results ought to be reported, in terms of choice of reportable attributes, and perhaps what supplementary material, such as explanations of attribute choice and methods or case studies of typical users, should be supplied. Choice of attributes comes first, and derives from the researcher's knowledge about the system's performance on the task and the customer's requirements; how to describe them is secondary.

[3] Earlier, we noted that writer and end-user are often the same: thus, all three user types are combined in the typical self-correcting word processor user of a checker.

However, it should be remembered that the Consumer Report is a multi-user artefact, in that it contains information relevant to a number of different customer classes, and indeed end-user and writer classes in which customers may be interested. It relies on self-diagnosis by the customer in choosing which attributes to pay attention to, and how to combine the values for those attributes to form their own evaluation. Thus, our aim is to choose attributes that will support most evaluations of grammar checkers. In terms of the functionality aspects we are concentrating on, this means producing information that can be interpreted for a number of different end-user/writer types.

The customer can be expected to know and construct their own comparative evaluation on certain aspects of end-user, writer, and system compatibility, such as the target language of the system and the language in which its advice is couched. Accordingly, such system characteristics will be included as straightforward attributes that will appear in the report.

We have motivated the production of a taxonomy of errors that is at least notionally couched in terms close to the writer's sources of error. However, it was also designed at quite a detailed level to support the development of test material, and so may be too detailed to be used as a set of attributes for reporting functionality to the customer. We may want to talk about the functionality performance for particular types of error individually — especially errors whose significance varies with writer or end-user classes. For other types, we may want to group a number together under a group attribute name. Our emphasis here is on NLP evaluation, and it is relatively clear that NLP, broadly conceived, is central to the functionality quality characteristic of grammar checkers. Given the mass-market nature of such tools, it is also clear that, whether the customer is the same as the end-user or not, it is likely that we will need to put a good deal of effort into finding ways of presenting and explaining the results of testing functionality in terms of attributes that are meaningful to the customer. Standard sets of examples and illustrations, and particularly case studies giving examples to allow a customer to diagnose their requirements, should be developed as part of our future work, and for anyone building on it, for example in a new language.

Thus, we have attributes based on error types and groups of error types. When we start to think about what sort of measure to use to convey performance on these errors, we encounter some complications that might lead us to convey some of the information not as measures (because they would be too complicated, or have too many dimensions) but as separate attributes, which might be grouped into aggregates or replicated individually for each error type attribute.

One such measure relates to *precision/recall*, which is a key property of this application, as in information retrieval, but here is complicated by the range of advice types. It is almost certainly not valuable to many customers to present measurements of recall separate from some consideration of precision (saying that a given system succeeds in all instances of *it's*/*its* confusion if in fact it flags all instances of either is likely to be misleading, even if the precision figures are given somewhere else). However, different end-users have different requirements and different interfaces may change the effect, as may customisability. An end-user who needs a lot of tutorial advice will not be benefitted by a high recall, low precision and poor advice type service, while they might be able to use the same recall and precision if the advice included enough information to let them make an accurate diagnosis.

Another such measure relates to the *type* of coverage a system has for a particular error type. Some systems may find only easily identified instances of an error type, but do that reliably. This may be of use to some users and not to others, who only make more complicated versions of the error. A measure that simply aggregates results of different levels of difficulty, with whatever weighting function, will not be able to support customer self-diagnosis on this dimension.

There is a case, therefore, for having separate attributes for noting false positive and coverage variability.

D.7 Methods

Assuming we have a list of attributes we want to report to the customer, we need to find ways of measuring these. The specification of methods is the definition of how to build up a value for a reportable attribute by combination of results of various component attributes, down to measurements that can be made directly, 'by inspection'.

The method has to define at least the following:

- Amount, composition and structuring of test material;
- Instantiations of interpretational scheme.

Concerning the amount of test material, the method should specify a critical threshold, i.e. the minimal amount of material for the results to be reliable and generalisable. In testing and evaluating natural language processing systems, we cannot obtain exhaustiveness in the linguistic coverage of the test material. A higher degree of completeness can, however, be achieved in types of phenomena to be tested than in tokens. Yet even as regards types, aiming for a complete coverage seems unfeasible in practice, which makes the issue of composition of the test material all the more important. The structuring of the material is rather a practical matter, as it has to do with systematising the material in a sound and transparent way, such that interpretational schemes can be readily applied to the results of the measures.

Here, we will concentrate on the composition and structuring of the test material.

D.7.1 Functionality requirement — part I

The detailed requirements outlined in section D.5 provide the backbone of methods for obtaining values for reportable attributes that deal with part I of the functionality requirement, which deals with positive coverage of errors. We illustrated those discussions with the example of English determiner-noun number disagreement, and we will continue with that example here.

Importantly, what was missing from those discussions was any indication of how the error would be represented concretely in terms of examples. Now that we come to test for coverage of that error, we must address the detailed description of the model of proofed text, which was introduced as an implicit standard relative to which errors are defined.

This part of the functionality requirement deals with positive coverage of errors — *recall* in information retrieval terms [4]. Our recommended method for measuring this is to associate with each error type a set of *error examples*. An error example consists of:

- A sentence with an error in it;
- The same sentence without the error;
- The name of the error type;
- Possibly, some indication of what forms of advice would be adequate to allow the correction process (for different user types — it should be clear how this relates to the criterion version of the requirements discussed earlier).

The rest of this section addresses the construction of the first two components of this structure. The advice types will be required in order to perform actual scoring of the systems' performance, but we will not consider them at this time.

[4] The issue of precision will be touched on in the next section.

The main requirement, clearly, is to generate a wide range of instantiations of the error type and its correction, varying in grammatical construction and complexity and in lexical choice.

In fact, we work 'back to front'. The corrected text examples can be generated using any test suite generator for NLP comprehension (cf. TSNLP work), or from suitable corpora, and so we start from there. We stated earlier, in our task model discussion, that error types should be defined in terms of transformations of proofed text examples — the examples given there are repeated here:

- OCR errors will be represented by typical letter or bigram substitutions in the end-product (literal output) of the proofed language model;
- Second language errors, such as wrong prepositions in prepositional phrases, will be represented as lexeme substitutions or possibly as subcategorisation errors if they are closely linked to verb frames;
- Concentration lapse errors or planning fault errors such as failed agreement between noun phrase determiner and head will be represented as the breaking of a constraint in the grammar of noun phrases.

Given such transformations, we can generate error sentences from correct text suite sentences. Each such pair forms the core of an error example. Note that different kinds of transformation operate on different levels of text structure, from character-level to syntactic constraints; the description of complex errors in terms of explicit transformations will be an important task for future work. The resulting test suite has a top level of organisation with a slot for each error type, which holds the set of error examples for that error type. Each set is subdivided by the various generation stages in which variations are produced, to provide a complete labelling for the source of each example.

This technique is purely *black-box*, relying on a model of the errors and corrections that is based purely on task analysis. There is a further set of techniques commonly used for testing grammar checkers that involves guessing plausible weaknesses in checker behaviour, which is more *glass-box* testing, relying on some ideas about the models of text used by the checkers. We do not deal with this at the moment, and it is possible that wide enough black-box testing will provide the same result. However, given the difficulties in automating the comparison of the advice given by a system and the advice needed by an end-user, it may be that truly huge test sets are much more difficult to administer for grammar checkers than, for example, for spelling checkers.

The method for using the error examples is relatively obvious: the error sentences are presented to the system, and we have to judge whether the system response would allow an end-user to produce the corrected sentence; as discussed previously the sometimes vague nature of advice given complicates this. Thus the error examples we have defined do not quite constitute a direct ('by inspection') method. The method will award a score to each response, with high scores for very specific correct advice graded through to very low scores for no advice or misleading advice. (For this part of the functionality requirement we do not penalise misleading advice more highly than no advice; that will be done in the testing for the second part, and the results combined as discussed in the following.)

Building on these scores, we then need to find a way to combine these test scores to get aggregate scores for the error type as a whole, or indeed for groups of error types where that is the level of reportable attributes. The weighting functions and the combination procedure constitute an *interpretation scheme* for the method, which is based on:

- The frequency of the error;
- The importance to the user (where that is different, as for particularly embarrassing errors!).

D.7.2 Functionality requirement — part II

The second part of the functionality requirement deals with what are commonly called false positives. Work on specific methods relating to this is still in progress. It is clear at this stage that it will involve the re-use (double interpretation) of test material developed for requirement part I, i.e. sentences containing error forms but also the corrected versions. Additionally, we anticipate the use of larger amounts of naturally occurring text to identify unexpected errors. Finally, it is anticipated that some use will be made of the technique of error-spotting (designing 'traps' for foreseen weaknesses of systems).

D.7.3 Methods for combination of attributes

In the section on reportable attributes (D.6) we mentioned that overall performance evaluations might want to include some combination of parts I and II, much as in information retrieval where recall and precision scores are combined to give a full picture of usefulness. We also discussed the complications inherent in deciding what reporting we would give to coverage variability, in the sense of whether certain identifiable subsets of linguistic coverage behave differently, and whether this is significant to users. Perhaps this sort of information should be present in our task model, in which case the example sets for the taxonomy of errors would be further marked with their identification in terms of these combination factors. (For instance, the various sentences for EDetNNum (see D.5.1) would be marked according to the complexity of the NP in which the error occurs.)

If we choose to present either of these kinds of information as separate attributes, we will require combination and weighting rules to determine how the direct results are to be used to arrive at the reportable attribute.

D.7.4 Composition and structuring of test material

The final test-kit includes at least a test suite, i.e. a carefully composed and structured set of inputs. The test suite will correspond to the set of error examples we have already discussed, as follows:

- For each attribute there is a horizontal tripartition: a set of erroneous inputs, a set of correct inputs and possibly a set of deliberately misleading inputs (setting 'traps' for the system);
- For each attribute there is a vertical ordering: the inputs cover the range from the most simple instantiation of a construction to the more complex realisations of it.

The correct inputs can be constructed according to general language rules or taken from excerpts of corpora, or be devised from a combination of these two. Likewise, the erroneous inputs as well as the deliberately misleading type of input can be constructed in the same manner. We do, however, stress that all erroneous or misleading input should be checked against non-proofed texts, and ideally supplemented with real-life occurences.

One obvious shortcoming of this method is its limited ability to control and substantially check the effect of interacting phenomena. To that end, a set of random sample texts (non-proofread real texts) must also be included in the test-kit.

D.7.5 Instantiations of the interpretational scheme

For instantiating interpretational schemes, collections of *spontaneous* and *non-proofread* natural language texts are a prerequisite. Note that this process is another way of looking at how our writer model is to be constructed.

To give a rough idea, the following collections are required:

1. A collection of texts from native speakers, possibly subdivided by:
    - Text type;
    - Educational level.
2. A collection of texts from non-native speakers, possibly subdivided by:
    - Text type;
    - Language (native language);
    - Educational level.

For a broad and general purpose instantiation of 1 to be applied to a grammar checker for native speakers, we would need a balanced corpus of texts to generate the rates for the frequency of phenomena. This is to be used for setting the weights and constructing the rating scales of the interpretational scheme for the measures. Along the same lines, we would get a similar instantiation for 2, but for non-native speaker grammar checkers for a language.

There is another function of such text collections; in the process of constructing the test suites they feed into these by providing examples and tokens of erroneous inputs to different categories. In fact, they may lead to definition of categories of error types.

Appendix E

Evaluation of Translators' Aids

This part explores evaluation of translators' aids in the spirit of the paradigms described in chapter 2, (sections 2.1.1, 2.2, and 2.6). All of its contents should be considered as 'work in progress'.

There are 4 sections, 3 of which refer further to a more elaborate appendix.

section E.1 examines *user profiles*. Dimensions of translation tasks and organizations involving translators are listed. In Appendix F, values on these dimensions are elaborated and typical examples described.

Section E.2 describes *types of translators' aids* that are actually present on the market. Appendix G lists and briefly describes products of 17 vendors of translators' aids.

Section E.3 *describes the functionality* of some typical translators' aids, esp. translation memories. Similar exercises are planned for other types of aid, e.g. term banks. It concludes with an introduction to actual feature checklists for translators' aids. Two such checklists are presented in Appendix H.

Section E.4 treats *evaluation procedure*. It lists some general requirements for validation and testing and applies these to measurements on translation memory.

E.1 Translators' aids: user profiles

Translation has traditionally been seen as the work of professional artisans who diligently translate other peoples' text in the best possible manner. Translation has been considered separate from other activities such as marketing or research and development - if not even subordinate to them. In some respects translation has been a buffer between the internal activities of an organization and its international contacts.

With the dramatic increase of international contacts the picture has changed. More and more people other than translators or interpreters are personally involved in translation and dealing with people in a foreign language, be it orally or on paper. The amount of text to be translated is growing and so is the number of languages involved. Furthermore, the level of translation that is expected is beginning to vary as information is sometimes needed more for its general contents than for reading pleasure. That is, the translation can be less than perfect in style. All this is starting to make a mark in the arrangement of translation activities in different organizations.

Thus, translation is no more just the activity of individuals but rather a collective process of an organization. Therefore the concepts of user and user profiles are used in this text to denote collective entities, that is organizations. Naturally the actual users of any translators' aid are individual persons, who could also be classified, but this perspective is given lesser notice in this text.

In the following sections, we give indicative summaries of the various dimensions relevant to distinctions amongst users. These summaries are not necessarily complete. They should be seen as starting points.

E.1.1 Dimensions of Translation

Translation activity can be described from several points of view or dimensions. These dimensions can be divided into dimensions describing the translated text, the translation organization and the external context of translation.

Translated text itself can be described with the following dimensions:

- quantity of translated text
- quantity of translation work
- text type
- characteristics of original text
- languages involved
- translation quality

The translation organization on the other hand can be described with the following dimensions:

- type of translation organization
- size of translation organization

Finally, the external context of translation in an organization can be described with the following dimensions:

- Nature of the entire organization
- Size of entire organization
- Amount of international activity
- Nature of international activity
- Language policy of the organization

Often some of these dimensions have more than one value, and these different values have varying weights. Therefore the aforementioned dimensions could most conveniently be considered as primary dimensions, which in their turn can be described by secondary dimensions. These secondary dimensions are statistical in nature:

- set of values
- frequencies of values
- average value
- variance of values
- minimum value
- maximum value

E.1.2 Relationships between different dimensions

The relations expressed below are far from being solid enough to serve as a standardization proposal. It is useful, however, to see how one can try to put some structure on the space of relevant dimensions.

- high ratio of exports → foreign language in common use
- subsidiary of international company → foreign language as official language
- high ratio of exports + small size → many foreign languages in common use
- quantity of work ↔ quality of translation
- quantity of work ↔ extent of translation work
- amount of int'l activity + size of organization ↔ quantity of translation
- text type → characteristics of original text

E.1.3 Trends in dimensions

- more subcontracting

- more multilingual organizations

In appendix F, we elaborate dimensions of translation; we also make a start with a user typology.

E.2 Product typology

For the sake of manageability and clarity, this section will contain only a basic typology of translators' software tools. Appendix G offers a list of commercial products which were available as of *March 1994*.

As computers evolve, new capabilities are becoming available, particularly with regard to inter-program communication (this is, among other things, making it easier to integrate specialized translation programs with existing software). The following classifications therefore represent a cursory snapshot of a rapidly moving target.

It is useful to note that most commercial systems currently available are essentially designed with a single user or a small group of users in mind. This causes problems when one thinks of applying them in larger translation services (see the annex on SdTK). A critical issue is *validation*. For a translation memory, for example, one does not aim for indiscriminate updating of the database with every new translation produced. Support for validation, even if only in the form of a 'layered' translation memory structure, is not found.

E.2.1 Multilingual dictionaries

A multilingual dictionary will typically consist of one or more bilingual dictionaries (one or both directions) and the prerequisite lookup (ie, retrieval) software. Most lookup packages are designed to accommodate more than one dictionary, although only one dictionary can be active at a time.

The retrieval software is generally designed to work in conjunction with wordprocessing software, although the specific manner in which this is implemented varies, depending on the operating systems.

A still rare but very welcome feature is inflectional morphology in lookup routines; this means that one does not have to enter a root form to retrieve an entry.

Several packages support supplementary and user dictionaries. Ideally, these are integrated in such a way that they are automatically searched at the same time the main dictionary is searched.

E.2.2 Multilingual thesauri

Relative newcomers to the translation world, multilingual thesauri typically consist of two or more monolingual thesauri cross-referenced by *concept* rather than alphabetically. By means of these links, a user can follow correspondences across multiple languages and rapidly browse an entry's subcategorizations of meanings and its synonyms. Like a traditional monolingual thesaurus, a multilingual thesaurus is best used to complement one's passive knowledge of a language, particularly as a memory aid for infrequently used words and for basic grammatical information, such as gender and inflection. As such, it is more useful as an aid for writing in a foreign language. For this reason, multilingual thesauri have found favor in the administrative sector among non-language professionals preparing correspondence in foreign languages. Professional translators, in fact, may find the basic vocabulary of the current generation of such packages too limited. Most translate *into* their mother tongue, of which they obviously have a higher active knowledge.

A well-designed interface can make an electronic multilingual thesaurus easier to navigate than traditional monolingual paper-based thesauri. In addition, the underlying lexical data can be organized in such a way as to allow other ways of navigating around entries than just on the basis of the formal synonym relationship. These *second order* relationships might include:

- associations
- antonyms
- hyper- and hyponyms
- part of
- derivations

One common enhancement to a multilingual thesaurus package is a notation facility for adding user entries. Another is a morphology-generation function, something particularly useful for the inflection-rich Romance languages.

E.2.3 Terminology management databases

Terminology management systems were among the first commercially available translation tools, at least in the PC world, and for this reason there is a larger variety available in this category than in other categories of translation tools.

A terminology management package will typically consist of a terminology database (*termbase*) and lookup software which allows users to consult the one or more termbases from within a word processing program. Certain packages consolidate both term-entry and term lookup in one program; others separate these functions. This is because an independent translator might want to compile termbases on-the-fly, while a central terminologist may prefer to distribute read-only termbases.

The file format structure employed in termbases is the subject of an ongoing debate in the terminology world. Many of the early packages adopted a strategy that can best be described as *structured text file*, while newer packages opt for a true relational model. In any event, today's term packages offer many of the facilities of a standard database system, such as security, validation, and even relational integrity.

A term package may also include external utilities for maintaining termbases, such as merging and reversing them, and for printing reports. Other utilities may include automatic term extraction from existing texts.

Several vendors also sell terminology lists in the given format of a term package.

E.2.4 Translation memories

The concept of a translation memory has been around for a long time—more than twenty years—but only recently has it become a significant commercial entity. Basically, a translation memory is a system which scans a source text and tries to match strings (a sentence or part thereof) against a database of paired source and target language strings with the aim of *reusing* previously translated materials. Some translation memories attempt only literal matching, ie can only retrieve the exact match of a sentence, while others employ fuzzy matching algorithms to retrieve similar target language strings, flagging differences. The flexibility and robustness of the matching algorithm largely determine the performance of the system, although for some applications (ie, highly repetitive material) the recall rate of exact matches can be high enough to justify the literal approach.

Translation memories are typically integrated into translation workstation packages, where they can be used in tandem with a terminology management system, a multilingual dictionary, and even raw MT output.

E.2.5 Miscellaneous

A number of other software packages do not fit neatly into the above categories. They include:

- Text alignment tools for generating translation memories and checking the quality of translations
- Corpus browsers for viewing translated strings *in situ*

- Morphology generators for quick lookups of inflections
- Terminology extractors for identifying new terms in texts

E.3 Featurization

A major precondition for assessing the quality of a software system is to define the overall functionality of the system, i.e. in our context to specify the features of translators' tools. Only if the functionality of a system complies with the individual user's requirements, will the software quality be judged positively. Therefore, featurization has to be based on

1 the knowledge of user requirements and
2 the knowledge of available translators' tools.

In this section, we describe the basic design and functionality of types of translators' aid. In the current version, this is only available for translation memory (section E.3.1). The section concludes with an overview of the two actual featurizations in Appendix H.

E.3.1 Design and function of translation memory

Any feature checklist in the context of evaluation needs to be standardized in the sense that it should be applicable for any such tool and the results should be independent of situational variables. Moreover, the relation between the availability of a certain feature and the quality aspect it denotes has to be defined carefully.

Therefore a featurization of translation memories should explicitly state the purpose(s) of each feature (component, function, or attribute) of the translation memory and give an explanation of how each feature serves that purpose. It should start from a description of the uses of a translation memory in translation.

To fix terms, we propose the following definition of a translation memory:

> *a translation memory is a multilingual text archive containing (segmented, aligned, parsed and classified) multilingual texts, allowing storage and retrieval of aligned multilingual text segments against various search conditions.*

Different translation memories differ as to the information stored along with the raw texts and the retrieval methods. This definition does not restrict translation memory to what is currently available in systems on the market.

The description below follows the division of functions in Steenbakkers' and des Tombe's featurization into off-line (analysis, import, and export) functions and online (in-translation) functions. These can be compared to a division of database functions into database management and database use (query/updating) functions in general. The point of this document is to bring out the theoretical questions, motivations, and options vis-à-vis each feature. Fuller featurizations are relegated to Appendix H.

E.3.1.1 Information contained

A translation memory is a collection of multilingual correspondences with optional control information stored with each correspondence. This characterization abstracts away from the actual manner of storing the correspondences (one-one, one-many, or many-many).

The control information can include information about the source text of the correspondence, its date, author, company, subject domain. This information may be used in ranking matches.

When a translation memory is used to support a given direction of translation, we can identify one segment of each correspondence as the (stored) source segment and another one as the (stored) target segment. A given query with a current source segment may return a number of correspondences with matching stored source segments.

E.3.1.2 Off-line function: import

Import transfers a text and its translation from a text file into the translation memory.

E.3.1.2.1 Import from raw format. Raw format is any format in which an external source text and its translation may be available for importing into a translation memory (ascii, word processor format, unsegmented, unaligned). Import from raw format may require preprocessing of the texts by the user outside the system and/or interactive editing of the text inside the system. The system may also be primed to accept texts in given external mark-up formats.

Checklist:
- input formats
- preprocessing
- post-editing

E.3.1.2.2 Import from native format. Native format is a format used by the translation memory program to save translation memory in a file. Native format may retain segmentation, alignment and control information.

Checklist:
- kind of native format
- format used in translation memory combinatorics (see section E.3.1.5)

E.3.1.3 Off-line function: analysis

Analysis may mean processing of a multilingual text before importing it into a translation memory, or processing of a monolingual source text before submitting it to translation, defining the input output relationship in each case. Analysis involves parsing of the source and target texts to some depth.

E.3.1.3.1 Textual parsing.
- Recognition of *punctuation* requires knowledge of the orthographic and grammatical conventions of a language, e.g. to distinguish end of sentence from abbreviation.
- *Markup* constitutes a kind of pre-editing. Increasingly, materials processed through translators' aid programs contain mark-up, as the translation stage is embedded in a multilingual document production line. A good analysis program has knowledge of different formatting conventions and mark-up languages.
- *Other special text elements* (proper names, codes, numbers, dates, currencies, tables, lists, figures) may be set off by mark-up. Parts of special elements such as tables, lists or captions may contain text that needs translation. Some special elements need not be translated, such as proper names and codes, others may need conversion to native format, such as numbers, dates, and currencies. Such elements are not always set off by mark-up.

E.3.1.3.2 Linguistic parsing.
- *Words:* base form reduction (lemmatization). Lemmatization or base form reduction is used to prepare (known or unknown) word lists and to prepare a text for automatic retrieval of terms from a term bank.

- *Phrases:* term or phraseology recognition. Syntactic parsing may be used to extract (known or unknown) multi-word terms or phraseology from a source text. Parsing is needed to normalise word order variation of phraseology (which words can form a phrase).

E.3.1.3.3 Segmentation. The purpose of segmentation is to choose the most useful translation units.

Segmentation involves a type of parsing. For simplicity, parsing tasks are usually arranged in increasing order of complexity with little feedback from higher level of analysis to lower. Segmentation is done monolingually using superficial parsing (punctuation, mark-up), and alignment is in turn based on segmentation.

It has been found that if translators manually correct segmentations made by the program, then later versions of the same document will not find matches against the translation memory based on the corrected segmentation, for the program will repeat its own segmentation (errors).

If we want to benchmark segmentation, we need standard, segmented corpora or a standard segmenter. But is there such a thing as 'correct segmentation'? What makes a segmentation correct? That it produces natural translation units: units that are of the right size for translators to work on, and which allow useful translation correspondences between source and target languages.

The unit must be small enough that their degree of repetition is sufficient to ensure a significant hit rate (the smaller the unit, the greater the probability of repetition). But the units must not be so small that their alternative translations vary too much (there must be enough context to fix the translation), or that they are proper parts of more useful translation correspondences.

The optimum size of translation unit may vary with text type and translation task. At one extreme, translation of a new version of a previously translated text may allow a large percentage of perfect or near perfect matches (only constants such as names and numbers may have been changed). At another extreme, different texts written in the same style, genre or phraseology may have few perfect matches but a high degree of repetition in the terminology and phraseology used. As phraseology in particular has no fixed string identity, recognition of such repetition requires deeper linguistic analysis.

An ultimate solution to the segmentation and alignment problems comes close to the problem of statistical machine translation: the task is to find alignments which give the best predictions for further translation correspondences.

In practice, translators proceed 97% of the time sentence by sentence, although the translation of one sentence may depend on the translation of the surrounding sentences.

Checklist:
- Input format
- Segment definition
- Output format
- Possibility of user intervention

E.3.1.3.4 Alignment. Alignment is the task of defining translation correspondences between source and target texts. Segmentation serves alignment which in turn serves the aim to increase the usefulness of the translation memory proposals for translation.

In principle, there should be feedback from alignment to segmentation: if translation uses different punctuation from the original, alignment may fail. A good alignment algorithm should be able to correct initial segmentation (put another way, alignment should consider other than 1-1 alignments between initial segmentations).

E.3.1.3.5 Term extraction. A further service that can be provided as an analysis stage is automatic (known or unknown) vocabulary or term extraction. Such extraction can have as input a previous dictionary or dictionaries. In addition, especially in the case of extracting unknown terms, it can use parsing and heuristics based on text statistics.

This raises a question of principle: what is the difference and division of labor between translation memory and term bank? Length of translation unit? Manual/automatic insertion of correspondences? Type, quality and amount of collateral information?

Users of TM/2 report that they use the term bank to store useful correspondences between segments not recognized by the TM/2 segmenter, not only well defined terminology and phraseology but other types of repetitive material as well.

E.3.1.3.6 Text statistics. Statistics is used to estimate the amount of work involved in a translation job. This is needed for planning and scheduling the work and for billing. Typical jobs for translation statistics are word counting and estimating amount of repetition in the text. Both tasks depend on the choice of unit of counting.

What constitutes repetition? Percentage of repetition depends on size of unit. Characters are trivially repeated; how large a percentage of individual words/sentences/paragraphs? What constitutes sameness (character per character identity, identity of base form of word)? What repetition is useful for translation?

E.3.1.4 Off-line function: export

Export involves transfer of text from the translation memory into an external text file. Import and export should be inverses.

E.3.1.4.1 Control information. What control information should be saved with a translation memory?

Should post-editing do search/replace in the aligned bilingual text, i.e. have access to the aligned text, perhaps even to the original translation memory response (mistakes being actually mistranslations rather than monolingual mistakes in the target text)?

E.3.1.5 Off-line functions: translation memory combinatorics

- Merging (Join)
- Filtering
- Inversion
- Composition

Think of translation memories as databases, and merging as database table join. A row in the table consists of a segment, its translation and control information. A join is based on identity or fuzzy match in one or more of the fields.

Merging happens in some systems in conjunction with analysis, where one of the inputs to merging is an input text. This is called filtering here. TM/2 has an analysis option 'Create a file of untranslated segments' intended for input to an MT system. This is a type of filtering too.

Inversion means reversing the direction of translation (exchange of source and target languages of the TM).

Composition means deriving a TM for the language pair A-C from TM's for language pairs A-B and B-C.

E.3.1.6 On-line functions

During translation, the main purpose of the translation memory is to retrieve the most useful correspondences matching the current source segment in the memory for the translator to choose from.

Desiderata on the online functioning of a translation memory include that the translation memory must

- show both the source and target text of the response
- indicate identities and differences between the query and the response
- evaluate (order) alternative responses against some (customizable) preferences
- allow easy insertion of (relevant parts of) the response into the translation.

For instance, in TM/2, preference is given to a match in the same document. This increases consistency within the text (e.g. several translators working with the same text will be given the same first choice).

E.3.1.7 On-line function: retrieval

Think of translation memories as databases, and retrieval as a database query. We have a partially described translation correspondence at hand and wish to retrieve from the translation memory one or more matching translation correspondences.

This symmetric way of looking at retrieval immediately suggests refinements of the usual query method. There could be many things the translators could restrict in the search condition besides the current source segment, in particular, properties of the target segment and control information.

E.3.1.7.1 Exact match. An exact match is a perfect character by character match between current source segment and stored source segment.

E.3.1.7.2 Fuzzy match. Everything else is a fuzzy match. Some systems assign percentages to fuzzy matches. Such figures are not comparable across systems unless the method of scoring is specified. The score may depend on the depth of analysis done to the source segment. A 90% match could mean 90% of the stored source segment is identical with the current source segment counting in terms of character strings, word forms, content words, or yet another unit meaningful for translation.

E.3.1.8 On-line function: updating

Here, it is useful again to compare translation memories with databases. Translation memory is updated with a new translation correspondence when a translation has been accepted by the translator. As always in updating a database, there is the question what to do with the previous contents of the database.

More generally, we can ask whether a translation memory can be modified interactively during translation by adding, deleting or changing entries in the translation memory.

Does the system allow one or several translation memories to be open simultaneously? If it does, can entries be transferred between translation memories? (Why should we want to do that?)

E.3.1.9 On-line function: automatic translation

A multilingual text archive lets the translation specify queries in the archive and returns responses for the translator to process as desired. A translation memory proper can do retrieval and even substitution automatically without help from the translator.

E.3.1.9.1 Automatic retrieval. An integrated translation memory in a translator's workbench features automatic retrieval and evaluation of translation correspondences.

E.3.1.9.2 Automatic substitution. Exact matches come up in translating new versions of a document. If you translate automatically, you don't get to check the translation against the original. Any mistakes in the original will carry over.

E.3.1.10 Networking

Networking during translation makes possible efficient translation of a text in parallel by a team of translators. Translations and term entries entered by one translator are immediately made available to others.

On the other hand, if terminologies/translation memories are shared before the translations are final, any mistakes made by one translator are broadcast as easily as correct translations. (On the other hand, consistently incorrect translations may be easier to fix in post-editing.)

E.3.2 Overview of feature checklists

Applying the notion of the Consumer Report Paradigm (cf. 2.1.1) to the overall process of evaluation, it is obvious that the starting point is to find out which are the relevant facts, features, tests and judgements in the context of translators' aids. There are two key elements in this process, i.e. the object of evaluation and the user (cf. section 2.2). Featurization in general, therefore, involves the integration of the user and the object aspects while at the same time considering the quality characteristics and attributes involved (cf. 2.1.1).

Carrying out user surveys according to the method recommended in section E.1 leads to a set of facts denoting the basic needs of users with respect to the environment in which the tools are applied. Carrying out product surveys (cf. section E.2) leads to a set of facts describing the basic functionality of the tools under question. Matching both sets of facts one arrives at a catalogue of facts relevant to users interested in these products. Such a catalogue of facts may range from e.g. the operating system (DOS/Windows/Windows NT/ UNIX/Sinix/Macintosh...) to the help facilities offered (context sensitive help/hypertext help/systematic help file...).

The next step in the overall process of preparing an evaluation involves assigning specific features to the facts which generally characterize such systems. The detailed nature of such feature lists mirrors the intention behind them, i.e. to elucidate those peculiarities of a software tool which might turn out to distinguish between tools of the same type. Feature lists are compiled in the awareness of the possibilities individual systems offer and with the aim of demonstrating the differences between similar tools.

In section E.3.2.1 a catalogue will be described, the goal of which is to assess to what extent a software tool fulfills the user's expectation/specification, i.e. the user's needs in general terms. In section E.3.2.2 we will describe what has to be considered when compiling feature checklists for translation memories and terminology management databases.

E.3.2.1 Specification/inspection catalogues

The specification/inspection catalogues were developed in the course of the ESPRIT projects 2315 and 6005 TWB (Translator's Workbench I and II)[1]. They are based on a previous user require-

[1] cf. Höge, M.; Hohmann, A.; Mayer, R.: Evaluation of TWB - Operationalization and Test Results. Final Report of the ESPRIT II Project 2315 Translator's Workbench (TWB). Stuttgart, 1992, unpublished and Höge, M.; Hohmann, A.; van der Horst, K.; Evans, S.; Caeyers, H.: User Participation in the TWB II Project - the First Test Cycle. Report of the ESPRIT II Project 6005 Translator's Workbench II (TWB II). Stuttgart, Paris, Luxembourg, 1993, unpublished.

ment study[2] which was performed in the framework of TWB I. In addition to user requirements, the catalogues which have been developed for term banks, translation memories, machine translation, machine-assisted terminology work, and checkers take into account the latest technical and functional achievements in the respective areas.

The catalogues serve as the basis for i) the specification of tools according to the needs of users and ii) the evaluation of to what extent the tool conforms with the user's needs or expectations. In general terms each catalogue comprises facts relevant to the software and related to a certain quality characteristic. Among the quality characteristics tackled in the ESPRIT catalogues are task adequacy, error tolerance, execution efficiency, ease of use, ease of learning etc. Users can tick items which are relevant to them, give them an individual priority and moreover specify this priority by rating its relative importance (0-100) compared to other items of the same type (target).

The specification/inspection catalogues also help to establish user profiles, since they demonstrate that the relative importance of particular items differs between different user groups, e.g. in the context of translation it is pretty obvious that a free-lance translator who works with a stand-alone PC will not give networkability a very high priority, while a translation department certainly will.

Having specified the relevant items and rated their importance, the answers can now be compared with the actual functionality of the software. The comparison between the previous specification and the actual functionality of the software works on the boolean basis of availability: yes/no. When an item specified beforehand is actually available, the full score will be given to the system, whereas no score will be given if the tool does not cover the item wanted. A numerical value, which can be calculated by comparing the target specification and actual functionality, demonstrates the percentage to which the software in general complies with the user's needs.

As 'facts' in the Consumer Report Paradigm are only the first step of the overall framework of decision taking, such catalogues have to be complemented by feature checklists (cf. the following section), tests and judgements (cf. section E.4).

E.3.2.2 Feature checklists of translators' tools

Any feature checklist in the context of evaluation needs to be standardized to make sure that it is applicable to any such tool and that the results are independent of situational variables. Moreover, the relation between the availability of a certain feature and the quality aspect it denotes has to be defined carefully.

Depending on the nature of the tool, only particular quality characteristics have to be considered for translation memories and terminology management databases respectively. Feature checklists should be based on the assumption that each feature mentioned in the list can be assessed and evaluated on the basis of

- a scale of good/bad
- according to the presence/absence of a feature
- the numeric value of the function/characteristic

The principal aim of any feature list is to deliver evaluation results without involving a large number of real users and extensive tests. The feature checklists are filled in by evaluators (probably with the assistance of real users) for a number of tools of the same kind. Thus the feature checklists for the different tools will be of help to real users who need some information on the systems under question. On the basis of the comparison of the different checklists, the user can finally choose the most suitable system for his personal environment.

A number of comprehensive feature checklists have been elaborated in the framework of EAGLES, i.e. checklists that focus on basic features relevant for a broad concept of term bank and translation

[2]cf. Fulford, H.; Höge, M.; Ahmad, K.: User Requirements Study. Final Report of the ESPRIT II Project 2315 Translator's Workbench (TWB). Stuttgart and Guildford, 1990, unpublished.

memory[3]. All feature checklists elaborated so far in the context of EAGLES need to be further integrated and harmonized according to the common framework, which will be an ambitious task for future work.

E.3.2.3 Further procedure - types and feature checklists

Having collated valuable work in this first phase of EAGLES, the primary concern of featurization in the next phase is to

- validate the catalogue of types as a first step in adequacy evaluation
- harmonize and synthesize the different feature lists
- validate the feature lists as a second step in adequacy evaluation

The main focus of the EAGLES evaluation/translation subgroup lies in providing worked-out checklists, mainly for translation memories and terminological databases. These checklists will be handed over to people active in the area of Linguistic Engineering, translators and system engineers. They will be asked to comment on the items and their relation to specific quality characteristics and to judge the usefulness of such checklists when it comes to buying one out of the plethora of tools available. Their comments will be considered and a final version of the checklists will be elaborated.

E.4 Evaluation procedure

In this section we take an initial look at evaluation procedures. In the end, the practical goal of standardization of evaluation is to make available some uniform and well-founded answers to the question *how to proceed* if one wants to evaluate a given object.

In the first general part, we list some general aspects of evaluation procedure. The second part applies this to complex measurement of translation memories. Similar treatments are foreseen for testing term banks and possibly other components of translators' aids.

E.4.1 General aspects of evaluation procedure

E.4.1.1 ISO on evaluation procedures

Source: ISO 9126: 1991, section 5.3. This section repeats to a large extent the summary of the ISO report already present in chapter 2.

The evaluation procedure consists of three stages and it may be applied in every appropriate phase of the life-cycle for each component of the software product:

E.4.1.1.1 Quality requirement definition. 'The purpose of the initial stage is to specify requirements in terms of quality characteristics and possible subcharacteristics. Requirements express the demand of the environment for the software product under consideration, and must be defined prior to the development. As a software product is decomposed into major components, the requirements derived from the overall product may differ for the different components.'

E.4.1.1.2 Evaluation preparation. 'The purpose of the second stage is to prepare the basis for evaluation.' This stage consists of three components:

[3] cf. Appendix H: Features of Translator's Workstations; descriptions were prepared for IBM TM II and Trados TW II.

1 Quality metrics selection: 'The manner in which quality characteristics have been defined does not allow their direct measurement. The need exists to establish metrics that correlate to the characteristics of the software product. Every quantifiable feature of software and every quantifiable interaction of software with its environment that correlates with a characteristic can be established as a metric. [...] Metrics can differ depending on the environment and the phases of the development process in which they are used. Metrics used in the development process should be correlated to the user respective metrics, because the metrics from the user's view is crucial.'
2 Rating levels definition: 'Quantifiable features can be measured quantitatively using quality metrics.' The result, the measured value, must be interpreted as a rated value, i.e. 'divided into ranges corresponding to the different degrees of satisfaction of the requirements. Since quality refers to given needs, no general levels for rating are possible. They must be defined for each specific evaluation.'
3 Assessment criteria definition: 'To assess the quality of the product, the results of the evaluation of the different characteristics must be summarized. The evaluator has to prepare a procedure for this, using, for instance, decision tables or weighted averages. The procedure usually will include other aspects such as time and cost that contribute to the assessment of quality of a software product in a particular environment.'

E.4.1.1.3 Evaluation procedure proper. 'The last step of the Evaluation Process Model is refined into three steps, namely measurement, rating and assessment.'

1 Measurement: 'For measurement, the selected metrics are applied to the software product. The result is values on the scales of the metrics.'
2 Rating: 'In the rating step, the rating level is determined for a measured value [...]'
3 Assessment: 'Assessment is the final step of the software evaluation process where a set of rated levels are summarized. The result is a statement of the quality of the software product. Then the summarized quality is compared with the other aspects such as time and cost. Finally managerial decisions will be made based on the managerial criteria. The result is a managerial decision on the acceptance or rejection, or on the release or no-release of the software product.'

E.4.1.2 Desiderata for testing methods

Tests for evaluation should, wherever possible, have the following properties:

- *reliable*: i.e. stable under repetition and under irrelevant changes of the context of the measurement like the person who applies it. There are well-known methods to establish reliability (e.g. split-half).
- *valid* i.e. the measurement values obtained inform us about the actual utility of the object of evaluation. As an example, one may try to infer, for some battery of tests, what the measurement results imply for the productivity of end users. In practice, a good aim is already the establishment of some correlation between measurement and productivity.
- *efficiently applicable*; especially, it is desirable that the measurements can be taken without involving users directly. Users do not want to be bothered by performing evaluations, they want others to do it for them. Users play a role in the *validation* procedure, precisely with the intention to eliminate them from later evaluations. However, validation will probably always remain somewhat problematic, so evaluations will probably always involve some degree of user activity
- producing values that are
  - *formal enough* to serve as a basis for comparison amongst alternative members of the class of objects of evaluation under consideration;
  - *mappable to utility* e.g., measuring the weight of some object of evaluation should only happen if it is clear how weight relates to utility; this is already a kind of a priori validity consideration

E.4.1.3 Test types

Attributes can be typed according to the kind of tests needed to establish their values for given objects of evaluation. Chapter 2 already described this.

In this section, we distinguish three main types of test. Test types differ wrt who does the testing (an evaluator or a translator), what data and tools are needed (program only or program and data, laboratory data or real data), and what the outputs of the test are like (quantitative or qualitative, objective or subjective-impressionistic).

E.4.1.3.1 Checklisting of features (= specification/inspection). A featurization (section 2.2) is a hierarchical feature structure describing an object: its components, functions, attributes and values. The featurization can be based on the manufacturer's data or checked out by the evaluator. Checklisting is done by the evaluator, it does not require complicated testing procedures. Checklisting is the method of measuring boolean (presence/absence of feature) and other formal-valued attributes.

E.4.1.3.2 Scenario test (= users test the complete system in a realistic environment). This involves putting the system into its intended use by its intended type of user, and recording the quality and efficiency of the work as well as user impressions. This is the natural point at which impressionistic evaluations are collected. (This is where this type of evaluation is most meaningful and reliable — first impressions can be deceptive, the real values will only emerge in continued use.) The testing is done by the translator, it involves real data, and the results can be qualitative, including questionnaires, impressionistic keyword evaluations (easy to use, slow), free-form reports, or quantitative (statistics).

Höge et al. distinguish comparative testing (comparing different products). Direct comparing of products can be the only way to obtain useful results for attributes whose values are only defined on a comparison scale. On the other hand, it may be difficult to obtain reliable results in comparative testing, in particular to guarantee that all the systems to compare are given equal attention.

E.4.1.3.3 Benchmark testing (= systematic testing using test tools and materials). A benchmark is a regimented test measuring a metric attribute. A benchmark should help determine objectively and reliably whether a function of a component achieves a given stated purpose.

Three ideas are central to benchmarking:

- benchmarking is standardization in the first place: a benchmark should be applicable to a class of objects and yield results of a sufficiently formal nature to be comparable.
- a benchmark has to be valid w.r.t. users' interests.
- a further desideratum for a benchmark is that it can be applied efficiently; ideally, it should not involve end users at all, but be applied by specialized benchmarkers and to a large degree be automated.

A benchmark is specified by specifying:

- the attribute(s) it measures
- the tools and data needed to run the benchmark
- the requirements on time, manpower and experience
- the procedure to follow to run the benchmark
- the interpretation of the results of the benchmark

Benchmarking translation memories involves quantitative tests based on specifically prepared materials on specific, central functions on the checklist. For instance, train a translation memory with a given reference text and run the translation memory against a source text, note the output: what and how many segments are found. This is done by the evaluator, it requires controlled data, and the outputs are quantitative.

From this, in an evaluation procedure guide a list should always be included of the critical functions to evaluate, measurable dimensions of them and ranges of acceptable values on them, and methods, possibly also tools to measure them. This covers what Höge et al. call task oriented systematic testing.

It seems that benchmarking is best suited for testing the quality characteristics of functionality/accuracy and efficiency.

E.4.2 Validation

Validation of tests ensures that the tests measure properties that have an effect on utility; in the case of translators' aids, important validation criteria are end user productivity and satisfaction. The customer's view may weigh these two differently from the end user's view.

A well-established method to establish the validity of some test is to rank systems according to test results and rank them a second time in actual practice and compare the rankings. This is slow and expensive. The comparison may be unreliable, both when the same subject tests the different systems and when different subjects are used, for different reasons. Also the effects of the different attribute-values and object components on the total efficiency are hard to separate.

There should be some more differentiating way to calibrate each component. Perhaps some way of tracing the behavior of the subjects (their utilization of each component - number of terms accepted from term bank, number of terms added to it, number of translations obtained from translation memory and saved into it, etc.).

Validation studies are not always of vital importance: sometimes, the 'internal validity' of an attribute is clear from the start, e.g. gas consumption of cars. For many software tools, speed seems to be of uncontroversial utility.

On the other hand, it is not really known whether e.g. a translation memory actually enhances end user productivity or satisfaction at all. Here NLP software is in a somewhat different state from the kinds of objects whose evaluations are found in consumer reports.

E.4.3 Benchmarking translation memories

Whatever the system, the choice of attributes to benchmark should be based on two practical considerations:

- what are important or central properties for product choice (usefulness)

- what can be measured validly, efficiently and reliably (feasibility)

The following sections apply these criteria to translation memories.

E.4.3.1 Usefulness

The bottom line of all translation aids is translation cost, which in turn depends on translation speed and translation quality. Both attributes are usually essential but inversely related; their relative weights may vary with the type of translation.

Translation speed in turn may depend not only on the speed of an individual translator, but the throughput of the whole production line, including pre- and post-editing, layout, terminology management, accounting, and so on. Using a system that allows a number of inter-communicating translators to work in parallel on the same job can achieve higher throughput than improvements on individual translation speed (cf. parallel processing vs. speedup of individual processors).

Thus we should describe different tasks and working setups for translators, and derive desiderata for a translation memory from them, and benchmarks from those desiderata. What is it that a translation memory is good for? The answers, or at least relative weights of different answers, may differ depending on text type and translation task.

For instance, there is a distinction between retranslation and translation of new material. When a document is retranslated (a new version of a manual, an instance of a fixed agreement text), large portions of the text may stay unchanged, or only constants (names, dates, codes, numbers) may change. Then the translation memory will contain previous version(s) of the document, and there is a high percentage of exact or near 100% matches. Parts of the document may be translated automatically and only changed segments presented to the translator.

At another extreme, a new text similar in style, terminology and phraseology to previous translations is to be translated. The percentage of exact matches to the previous translations may be low, but there may be significant overlap in style (terminology and phraseology) that should be exploited. The system should have an intelligent fuzzy matching algorithm or the search of useful matches should be left to the user.

A text may exhibit significant repetition internally. In this case, the translation memory should be updated as the translation proceeds so as to exploit the autocorrelation and ensure consistency throughout the document.

To sum up, a translator needs a translation memory to:

- avoid redoing translation of repeated material
- use previous texts as a model for new translations
- ensure consistency throughout a translation

What are the most important properties of a translation memory given these objectives? Note that some properties are interdependent and can be inversely related (e.g. hit rate and speed of retrieval). The properties can be divided into off-line and online properties, alias population and use of a translation memory.

In general, much of the work to ensure useful matches during translation time can be done off-line, insomuch as it depends on materials available ahead of translation (model texts, source text).

For example, the size of the online translation memory can be kept proportional to the size of the text being translated, by filtering a larger translation memory (or several of these) against the text to be translated, so as to obtain a new translation memory that only contains the matches found (possibly with enough surrounding context to help gauge the relevance of the match).

Autocorrelation forms an exception to this, i.e. the case where the translation memory is updated as the translation proceeds and used immediately in translating the rest of the text.

E.4.3.2 Online properties

E.4.3.2.1 Size (capacity of translation memory). This has an effect on hit rate and speed. The optimum size is likely to depend on text and translation type. The less repetition, the more text is needed to obtain an acceptable hit rate. (Again filtering may help.)

E.4.3.2.2 Speed (retrieval time). This depends on size and on the match algorithm. In translation in general, the raw speed of translation aids is strongly dominated by the time spent by the user in screening and editing translation proposals. What may be of interest is the behavior of the system (speed, hit rate) as the size of the memory grows (to determine optimum translation memory size).

E.4.3.2.3 Hit rate (number of matches). This depends on size, segmentation, match algorithm, and match evaluation algorithm. There is an optimum to shoot for here rather than a maximum. In general, translators prefer few high quality matches to many matches of doubtful use. The reason is simple; screening matches by eye is slow, tiring, and error prone, and editing around a bad match may be slower than writing a new translation from scratch.

E.4.3.3 Off-line properties

E.4.3.3.1 Text analysis methods. We would like to stress the quality characteristic of customizability or adaptability. Systems currently on the market tend to have an overly narrow view on text analysis: built-in parsers for segmentation, built-in match algorithms. At least in one case, no off the shelf translation memory was found acceptable because segmentation could not manage sentences of over 100 words in length and did not provide any easy way to modify the definition of segments. The SGML style approach with a generic parser which can accept user defined grammars should be pushed as a standard here as well.

Another aspect is depth of linguistic analysis. This cuts two ways: systems which depend on linguistic analysis become language bound. For instance, TM/2 is not easily extended to a language like Finnish with rich morphology. On the other hand, future better (more human like) fuzzy matching algorithms are bound to identify units meaningful for translation (such as shared words, terminology, phraseology, or similar grammar).

E.4.3.4 Segmentation and alignment: success rate

These affect hit rate and usefulness of matches. What should count as "correct" segmentation and alignment depends on the usefulness of the resulting matches to the translator. That can depend on text and translation type. Given a particular definition of segment, we can benchmark whether a translation memory program segments and aligns a text and its translation according to the definition (accuracy). Or, we can try to measure the usefulness of a given method of segmentation or alignment for translation (suitability). See section E.4 for further discussion of these types of benchmark.

E.4.3.5 Initial (raw) import into translation memory

This measures portability. This is of interest to translators that have a large quantity of translated material at hand before acquiring a system, or to translation companies that work with a variety of systems. Here, EAGLES could propose or endorse a standard format for aligned texts (e.g. one defined by the TEI initiative).

E.4.3.6 Translation memory combinatorics

These interact with size and speed too, because good off-line methods for constructing text specific translation memories can diminish size and improve hit rate during translation. The possibilities of inverting and composing translation memories may not be as practical as they may seem (in general, translation relations are not symmetric nor transitive).

E.4.3.7 Export of translation memory

This may be of particular interest for loose knit work groups using various types of tool/environment and communicating irregularly and/or via slow connections (modem, fax, diskette, paper).

E.4.3.8 Specifying benchmarks

Starting from the feasibility angle, to define a benchmark we have to be able to provide:

- the attribute(s) it measures
- the measures used
- the tools and data needed to run the benchmark
- the requirements on time, manpower and experience
- the procedure to follow to run the benchmark
- the interpretation of the results of the benchmark

Attributes have been discussed in the previous section. Some *measures* one can think of are:

- number of revisions for a given translation (measures translation quality)
- number of keystrokes during the performance of a given task
- mean time to failure during the performance of a given task
- presence/absence of functions listed on a checklist
- time needed for given task
- number of errors made by user
- number of errors made by component
- overgeneration by component (number of items)
- number of unsuccessful communications user–component

The inputs of the benchmark can be any of the following:

- raw input
- test corpus
- test suite

The differences are these: using raw input, the tester has to check all instances of the input-output relation, to ensure that when the input is of a certain type, then the output is of an expected type. Using a test corpus, the tester can assume that the inputs are of a certain form, and it suffices to check that the outputs satisfy expected criteria. Using a test suite, it suffices to compare the outputs of the benchmark to the standard outputs of the test suite to find any differences.

Test suites make sense when the range of acceptable outputs is not too varied (for instance, there is only one correct output for each input). This is not generally the case for translations: there is no one correct translation of a given segment. Individual benchmarks may still satisfy this restriction: e.g. a benchmark checking retrieval of exact matches could input a corpus of n distinct segments and check that these segments are retrieved against the same corpus as the source text.

The next question is how the data and/or tools used in the benchmark are provided. There are various alternatives:

1 The benchmark contains specifications of the input (data and tools) needed. The benchmark tester provides the input and checks that it satisfies the specifications by whatever methods seem fit.
2 The benchmark contains specifications of the input plus tools to check that the input meets the specifications.
3 The benchmark contains tools to produce the kind of input needed.
4 The benchmark comes complete with the kind of input needed.

Alternative 1 imposes all the work on tester, while 4 is easiest for the tester. On the other hand, 4 is least flexible and leaves the door open for 'tweaking' the system with benchmark specific optimizations. Alternatives 2-3 seem best in principle: they provide an unlimited supply of data as well as objective control on the quality of the data.

There is a major watershed between benchmarks that are run between the tester and the program and those that involve users as test subjects. In fact benchmarks that involve users threaten to come close to scenario testing. If there is a difference, a benchmark involving users would just quantify limited aspects of user behavior (keystrokes, time, errors) in a limited test situation, trying to control variables extraneous to the particular benchmark. A scenario test would attempt to simulate a normal working environment.

The interpretation part of the benchmark specification should tell a tester what the benchmark can tell about a product. For instance, if it is critical for a site to have access to a large mass of texts simultaneously, run a capacity benchmark. If there is a high turnover of personnel or a deadline to meet, run a learnability benchmark.

E.4.3.9 Suggestions for translation memory benchmarks

E.4.3.9.1 Translation memory: segmentation and retrieval. Component: translation memory, application.

Related to attribute: accuracy (complete, correct)

Procedure:

1 Do the following for various settings of 'fuzzy match'
2 Define some class T of texts (text type *times* subject field)
3 Take an empty translation memory
4 Fill it on the basis of some subset of T (note that the filling/updating facility is not the component to evaluate)
5 Apply it to other members of T
6 Score for percentage translated, and percentage translated correctly (in the new context)
7 Compute scores based on notions like recall and precision

The results will depend a lot on the variability within T but that will not prevent them from serving as a basis for comparison amongst translation memories. Part of the experiment is automatable, for other parts only limited human intervention is necessary (providing input for memory filling, hand-checking translations generated from the memory).

The method described above is applicable for evaluating the system's ability to find 100% matches. (Only there the notion of percentage of match is well defined.)

A further refinement is to study the usefulness of (less than 100%) matches. This could be done empirically by observing the use translators make of (non-100%) matches.

Another more principled approach is to use calibrated texts, using system independent methods to determine useful matches in them.

Instead of providing ready made texts, it would be more useful to provide methods to produce such, i.e. for evaluating the degree of repetition in a given text. This angle has some theoretical interest too for translation theory. EAGLES should review literature on this.

E.4.3.9.2 Translation memory, import from external format. A similar benchmark is possible for translation memory in update mode. Given some *training base* T (a set of predefined translations), measure the effort involved in executing the updating for T.

E.4.3.9.3 Translation memory, learnability. Benchmarking ease of learning and using is a lot more complicated. A problem is that most of these require simulated actual use, and so involve a human user.

This makes the measurements less reliable and moreover less efficient to apply.

For example, think of the idea of benchmarking learnability of translation memory use. Typically, such a benchmark will be defined as follows: define some initial state q_0 of a human being, e.g. a translator who has no experience with using translation memories at all yet. Define a criterion state q_c which is the state of this same person but now a competent user of the system. The experimental task is to get from q_0 to q_c. Measurement is e.g. the time needed and the number of keystrokes applied. The problem is that after having done this for system A, our experimental person is no longer in state q_0 and so we need somebody else for the measurement on system B. But people differ in all sorts of relevant aspects like intelligence, so the measurement is not automatically reliable. Normally one would try to compensate by applying various conditions to randomly composed groups of people from a rather homogeneous population; we can propose that but the idea of *efficient* benchmarking suffers a lot.

E.4.3.10 Translation memory, portability

For some other attributes of interest, like portability, benchmarking may come down to comparing systems with a pre-given checklist.

E.4.3.11 Benchmarks suggested by Steenbakker's featurization

In the following, 'primitive actions' are actions like pressing a key on the computer's keyboard or clicking or dragging with a mouse.

Determining the minimum number of such actions needed for some given task may be a measure of 'user-friendliness'. Of course the possibility to define macros has to be taken into account.

1 Segmentation
   average speed
   average percentage of correct segmentations
2 Alignment
   average speed
   average percentage of correct alignments
3 What is the minimum number of primitive actions for performing alignment of two given texts?
4 What is the minimum number of primitive actions for importing an aligned source and target segment into a translation memory?
5 What is the minimum number of primitive actions for adding unaligned source and target segment into a translation memory during translation?
6 What is the minimum number of primitive actions for performing different translation memory management functions?
7 What is the minimum number of primitive actions for performing the different tasks involved in translating a source text in translation memory mode?

We propose to choose as an experimental benchmark the ability of a translation memory to recover different sorts of near (fuzzy) matches. The test will consist of entering a given text to the translation memory, performing certain systematic changes on the text, and measuring the recall of the system after those changes. The changes will range systematically from identical segments to changes of punctuation, changes in constants (numbers, names), changes in segment length (shorter and longer segments than stored), changes in wording (words substituted, added or left out), changes in sentence structure (word order, grammatical construction), etc. The tests will be so designed that they would give an indication as to what kind of fuzzy matching algorithm the system is using.

The test material can be a test suite (a given text with a set of variants), or, even better, a program that will make various types of change to any given text. This would appear to be one of the most crucial types of benchmark one can run on a translation memory, one that gets to the heart of this type of program.

Appendix F

User Profiles

F.1 Dimensions of translation

F.1.1 Quantity of translated text

The most obvious dimension of translated text and translation activity in general is the quantity of translated text. It is also easy to quantify and measure. Quantity can be measured in units of pages or words or even characters, which are all related to each other. Sometimes text is also measured in lines. Pages consist of words which consist of characters. Naturally the number of words per page depends on the definition of a page, for instance the size of the page and the size of the font. In this text a standard page is understood to contain 2000 alphabetic characters. Section F.1.10 contains a sample of different standards of measurement of the quantity of translated text in five European countries.

In addition one has to note that in different languages the average quantity of words per page varies according to the properties of the language. For example, a text in Finnish usually adds up to fewer words than the equivalent text in English. One reason for this difference is that Finnish uses inflection where English uses prepositions. A point to remember is whether one is measuring the quantity in the source language or the target language. Normally the quantity is measured from the source language

Another difference in the measurement of quantity of translated text exists between idiographic languages such as Chinese and Japanese, and alphabetic languages such as European languages. In languages with an alphabet, an individual word is composed of one or more characters, that is letters of the alphabet. On the other hand, a substantial number of individual words in idiographic languages correspond each to an individual character or idiogram. Thus a text of X characters in Japanese may correspond to Y words in English

The quantity of translated text can be measured text-by-text or on a time scale, usually on a monthly or yearly basis. Quantity measured text-by-text can vary greatly, but usually there is clear minimum and maximum and naturally an average quantity. Quantity measured by year gives a general picture of translation activity within an organization. This figure can vary from several hundred pages to tens of thousands of pages a year, depending on the organization.

F.1.2 Quantity of translation work

The volume of translated text can also be described by how much time has been used in the translation work. Usually this is measured in man hours. Naturally the amount of work correlates with the quality of the translation, which may vary from text to text. In addition, the amount of work depends on the target and source languages.

F.1.3 Text types

Text type defines firstly what is the purpose of the text. General classes of text types are the following:

- Correspondence
- Journalism/Communication
- Business/Commercial
- Marketing
- Advertising
- Administration
- Legal
- Scientific
- Technical
- Culture
- Literature

These classes are not mutually exclusive. For instance, advertising can be considered a subclass of marketing, and marketing a subclass of business. In addition, organizations are not restricted to one text type.

Text type also partly indicates what sort of terminology the text includes. Terminology can be general, field specific or company specific. In its own right, terminology can be classified endlessly. Under the type of technical text, different fields of terminology could be for instance:

- electronics
- telecommunications
- mobile phone technology

Text type and terminology vary from text to text. Every organization uses normally several text types. An essential secondary dimension of translation activity is the set of different text types and their respective frequencies within an organization. Possibly some text type can be dominant, or it may be possible that there is a large set of text types of equal frequencies. Another secondary dimension along the same line is the set of different special sets of terminology that the organization uses and their respective sizes. The size of a set of terminology is understood here as the number of individual terms grouped under the set of terminology. Both of these could be described qualitatively as being between

- narrow
- broad

F.1.4 Characteristics of original text

Characteristics of the original text describe qualitatively the style of the text. This is partly a subjective dimension, but can also be more objectively described with diligent analysis of the original text. It is also a relative dimension, since it can only be measured against other texts. Thus it is dependent on text type for instance. At least until now there exist no quantitative universal norms or standards for characteristics of text. Typically the original text can be characterized as being:

- repetitive
- idiomatic
- terminology-rich
- terminology-poor

Repetitiveness means that the text contains more repetitions of individual terms or words, phrases, sentences, and even full paragraphs than usual. It may mean word-for-word repetition or similarities in constructions used in the text. Thus repetitiveness may also imply that word order or the structure of sentences is more consistent than usual. Typically, repetitive text is encountered in technical text types.

Idiomatic is in many respects the opposite of repetitive. Basically idiomaticity means the use of the expressive possibilities of the language to the fullest in the text. It may imply a large variance of words or the use of rare words and colourful idioms. This may lead to the use of many different terms for basically the same concept for the sake of nuances. Idiomaticity may also imply greater variance of sentence structure and other constructions than usual. Typically, idiomatic text is encountered in literature and advertising.

Terminology-rich means that the text contains more specific terminology than usual. Specific terminology is understood here as words and phrases being used only within a specific field or having meanings deviant from general vocabulary. Thus understanding terminology-rich text requires knowledge of the specific field in question. Typically, terminology-rich text is encountered in scientific text of any kind and technical text.

Terminology-poor is not exactly the opposite of terminology-rich. It can be understood to mean that the text simply uses only general vocabulary and thus should be understood by anyone with good general knowledge of the language. Terminology-poor can also be understood to imply that the language lacks necessary standard terms in some specific field and thus has to use alternative expressions or constructions. This may lead to long and awkward sentences which may contain ambiguity or lack exactness.

F.1.5 Languages involved

Translation work usually involves many translation directions. A translation direction is a pair of languages, of which one is the source language and the other the target language of the translation work. The source language is the language of the the original text from which the text is translated. The target language is the language into which the text is translated. The direction of the translation is not negligible, since the ability to translate in one direction does not necessarily imply the ability to translate in the opposite direction. Very often the target language is the native language of the person doing the translation work.

Typically some translation directions are more important than others. Effectively this dimension can be described by the set of translation directions and the respective quantities of text translated in each such pair. This dimension determines what language skills are needed in translation work. It may also help determine how the translation activity should be arranged by the organization.

Normally the translation directions are not random. Often there is at least one language which is a major source language or target language or both at the same time (though the European Community is an example of a truly multilingual organisation without, at present, focal *source* languages). Such a language will be called a focal language in this text. Usually the focal language is the national language, but there may be one or two others. It is even possible that the national language is not a focal language at all. Typically focal languages are used as intermediary languages when a translation is needed to or from a rare language. In such a case, English is the focal language par excellence.

Typically organizations have up to ten or slightly more translation directions, 2-4 language pairs, and 1-2 focal languages. If translations concern more languages, the translation is usually done via an intermediary language in which the organization has the necessary language skills. An exception is the European Community's SdT, with a much larger number of language pairs dealt with directly. See the annex on this institutionK).

F.1.6 Translation quality

Until recently, there has basically been only quality translation, meaning the best a translator could reasonably offer. Translation work has strived at perfection and nothing less. Nowadays, however, demand has arisen for variation in both directions. Levels of translation quality can be described at least in the following terms:

- raw translation
- normal quality translation
- extra-quality translation
- adaptation of original text

Raw translation means a translation which conveys the central meaning of the original text. There may be grammatical errors and misspellings, but the text has to be understandable. Typically, this could be translations of large amounts of scientific abstracts.

Normal quality translation corresponds roughly to the translations of old. The original text is translated fully and the translated text is grammatically correct and reasonably fluent. The text may be awkward at times, but the contents of the original text should be understood completely from the translation. Typically, this could be a translation of a technical manual.

Extra-quality translation implies that the translated text is both fluent and idiomatic. The translation should be assimilated completely to the cultural context of the target language. One should not be able to recognize the translated text as a translation. Typically, this could be an advertisement brochure or a piece of literature.

Adaptation of original text is not actually the direct translation of text but the production of new text based on foreign language original(s). The resultant text need not have to correspond sentence by sentence to the originals, but may instead even have omissions or reorderings according to what the translator deems appropriate. The resultant text is expected to be fluent language.

Section F.1.11 contains a sample of different classifications of quality in use in five European countries. As quality of translation has started to vary so has the extent of translation work. In addition to merely translating texts translators are being required to handle new tasks both before and after the translation work. The following classifications can be made:

- proof-reading and grammar checking
- updating old translations
- making new translations
- editing and translating text
- editing and translating text and making the layout
- oral gisting: producing a quick oral summary of the content of the source text

Proof-reading and grammar checking basically means going through a translation done by another person for possible errors and stylistic reasons. This person may vary from a professional translator to an ordinary employee. The length of this work varies naturally with the level of quality of the translated text, which in turn depends on the linguistic skills of the original writer. Typically it should take considerably less time than the actual translation work, but in some cases it may lead to rewriting the translated text completely. Actually some checking of the source text is sometimes done and may serve to simplify the subsequent translation task.

Updating old translations means translating only those parts of the text which have changed in the original. This requires that the effects of the changes are such that translating the new original in full would take considerably more time than just translating the changes. It also implies that the changes are relatively easy to identify. This type of translation work is typical especially for technical manuals. Version control is an important problem in this area.

Making new translations means doing the translation from scratch. Different levels of doing this have been described above. In addition to merely translating the text, translators are often expected to raise the quality of the text by editing the resultant text. Furthermore, translators are sometimes also required to take care of the layout of the resultant text. These auxiliary tasks are on the increase and adding to the skills required of translators.

Naturally the level of translation quality corresponds directly with the quantity of translation work. So does the extent of translation work.

F.1.7 Dimensions of the translation organization

F.1.7.0.1 Type of translation organization.
- translation company
- bi/multilingual organization
- centralized translation activity
- decentralized translation activity
- subcontracted translation activity
- combination of above

What is the background of people doing the translation?

- full-time professional translators
- editors
- domain experts
- non-translator support personnel (secretaries etc)
- combination of above

F.1.7.0.2 Size of translation organization. Amount of employees

- full-time translators
- part-time translators
- support personnel

F.1.7.0.3 Nature of the entire organization. What is the business of the company?

- domestic company
- domestic public organization
- subsidiary of an international company
- international company
- international public organization

F.1.7.0.4 Size of the entire organization. What is the amount of employees? What is the turnover of the company?

F.1.7.0.5 Amount of international activity. What proportion of the employees is directly involved in international activities? What is the share of exports of the whole turnover?

F.1.7.0.6 Nature of international activity.
- Export
- Import
- Localization
- General

F.1.7.0.7 Language policy of the organization. What is the official language of an organization?

- national language as official language
- foreign language in common use
- foreign language as official language
- several languages other than national language in common use
- multilingual as a matter of principle

F.1.7.0.8 Typical profiles.

F.1.7.0.9 Free-lance translator.

- quantity of translated text: X pages a year
- quantity of translation work: up to 1 man year, often part-time
- translation quality: high
- languages involved: single focal language
- size: 1 person

Aids used in decreasing probability:

- manual dictionary
- word processor
- word lists

F.1.7.0.10 Translation company.

F.1.7.0.11 Organization with centralized internal translation department.

F.1.7.0.12 Organization with subcontracted translation.

F.1.7.0.13 Organization with decentralized translation.

F.1.7.0.14 Bi/multilingual organization.

F.1.7.0.15 International organization.

F.1.7.0.16 Organization with hybrid profile.

F.1.8 Correlations between user profiles and translation aids

Compilation of translators' aids mentioned in section E.2.

F.1.9 User case examples

F.1.9.0.17 Translation Company: Transbus/Sweden. "Swedish mid-sized translation company"

Status: questionnaire received with feasible answers, requires one telephone interview. In reserve, I am waiting for response from Donnelley Language Services/Netherlands.

F.1.9.0.18 Organization with internal translation department: LM Ericsson, Sweden. "Large Swedish electronics company"

Status: questionnaire received with feasible content, requires one telephone interview. In reserve I have the response from Bull/France, which has feasible content.

F.1.9.0.19 Bilingual organization: Vaisala Oy, Finland. "Mid-sized Finnish electronics company"

Status: questionnaire sent, response promised by week 15. In reserve I have their responses to an earlier questionnaire.

F.1.9.0.20 International organization: WHO. "Large international organization with headquarters? in Switzerland"

Status: I am cooperating on this with Olivier Pasteur. No reserve source here.

F.1.9.0.21 Public organization: Swiss Chancery. "Swiss public organization"

Status: Questionnaire received with feasible content on April 8th. May require a short telephone interview.

F.1.9.0.22 Free-lance translator: Canadian/Quebecan survey. Status: Report received April 8th. If the report proves inadequate, I will try to supplement/complement it with interviews here in Finland with the help of Lauri Carlson. There might possibly be a similar report made in France.

F.1.10 Measurement of translation quantity

Different standards of measurement of the quantity of translated text in four European countries:

- France
- Germany
- Switzerland
- Denmark

Status: Questionnaires sent, answers have been promised by week 15.

F.1.11 Measurement of translation quantity

Different classifications of the quality of translated text in five European countries:

- UK
- France
- Germany
- Switzerland
- Denmark

Appendix G

Translation Tools

G.0.12 Introduction

This appendix contains a listing of translation tools arranged by vendor. The criteria for inclusion are:

1 packages are commercially available, ie not research prototypes
2 packages were available as of 1 March, 1994
3 packages are known to compiler of this list

It will be apparent that the following packages do not necessarily fit neatly into the categories defined earlier in this report.

G.0.13 ALPNET

Product

TSS

Comment

Nearly ten years old, the ALPnet package TSS (OS/2) does not truly satisfy the first criterion of being "commercially available." While the Utah company actively promoted its software in the past, ALPnet currently prefers not to attract additional customers for the package. However, the company supports its several large corporate customers and it continues to use TSS internally within the ALPnet translation network.

TSS is an integrated translation system with one of the first implementations of the translation memory concept. A function called AutoTerm automatically looks up terminology in the system's termbase.

Vendor

ALPNET
4444 South 700 East, Suite 204
Salt Lake City, UT 84107-3075
USA
Tel: +1 801 265 3300
Fax: +1 801 265 3310

G.0.14 CAP debis

Products

Keyterm, Keylex for Windows

Comment

Keyterm is a terminology database which runs under Unix. Based on the Informix RDBMS, it is a fully relational system and is concept- rather than lexeme-oriented.

Keylex is a Windows-based retrieval package which works in conjuction with Keyterm. Designed for viewing Keyterm termbases, it lacks the Keyterm's extensive termbase building and management facilities.

CAP debis is a systems integration house, and it is not entirely clear how the company intends to market its terminology packages and standalone commercial packages.

Vendor

CAP debis Systemhaus KSP
Geb. 41.0/LDYKSP
München, D-81663
GERMANY
Tel: +49 89 607 32 152
Fax: +49 89 607 29 968

G.0.15 Canadian Workplace Automation Research Center

Product

Translator's Workstation

Comment

Strictly speaking, the CWARC Translation Workstation (DOS) does not satisfy the first criteria above in the sense that you cannot actually *buy* it anywhere. However, it is included in these listings because a) there is some confusion in the translation world as to what it exactly is, and b) it is intrinsically interesting.

Initially, the CWARC, a group of Canadian researchers, designed an integrated translation system based on off-the-shelf commercial DOS packages (WordPerfect, MTX, the Termium CD-ROM, Verbatex, and CompareRite) and integrated them within the DOS multitasking system DESQview. CWARC has since added additional modules, including network capabilities, Le Robert Electronique, and Naturel, a full text retrieval system.

CWARC is planning to design a Windows version of the Workstation. While CWARC may also eventually add custom software modules which it has developed itself, the primary exercise was to create a translation tool suite based on existing software.

CWARC presents the Translation Workstation as a model, rather than a package.

Vendor

Canadian Workplace Automation Research Center
1575 Chombley Boulevard
H7V 2X2 Laval, QUE
CANADA
Tel: 1 514 682 3400
Fax: 1 514 686 1990

G.0.16 Collins

Products

Collins Online, Collins Series 100

Comment

Collins Online is a series of six bilingual dictionaries (DOS, Windows) based on British publisher HarperCollins's Collins Gem pocket dictionaries. Online is based on the LinguaTech (see LinguaTech) retrieval software and allows users to set up both supplemental and user dictionaries.

Collins Series 100 (DOS, Windows) is a new line of four bilingual dictionaries based on the more extensive Collins College paperbacks. Series 100 employs the CompuLex retrieval software from AND Software.

Vendor

Collins Electronic Reference
14 Steep Lane, Findon
Worthing, Sussex, BN14 0UF
UK
Tel: +44 903 873555
Fax: +44 903 873633

G.0.17 GlobalWare

Products

XL8 Code, XL8 Help, XL8 Documentation

Comment

Introduced in 1992, XL8 Code (DOS) is an interactive tool for translating and maintaining user interface text strings in C language source code. It is particulary strong in reusing previous translations and automatically handling revisions; its fuzzy matching algorithms scan new source language files for previously translated materials and make substitutions where possible. While it runs under DOS, XL/8 supports character sets from other platforms, including Windows, Macintosh, and Unix.

In early 1994, GlobalWare brought out complementary packages called XL8 Help and XL8 Documentation. The former is designed for translating help files, in particular the intricately structured help files of Windows programs; the latter is for translating product documentation. Used in tandem, the XL8 programs offer the possibility of standardizing terminology and user-interface text strings across the source code, help files, and documentation of a software package.

XL8 was originally developed by Los Angeles translation company IDOC for internal use. It later set up a separate, GlobalWare, to market the package commercially.

GlobalWare
10474 Santa Monica Blvd, Suite 304
Los Angeles, CA, 90025
USA
Tel: +1 310 441 1024
Fax: +1 310 441 0824

G.0.18 IBM Deutschland

Product

TM2/2

Comment

An integrated translation package (OS/2) which has explicit support for nineteen source languages and upwards. It can handle any target language currently supported by 0S/2. TM/2 has facilities for bilingual dictionariest. In order not to compete with dictionary publishers IBM does not supply these.

TM/2 offers a programming interface (API) for use in conjunction with an MT system.

IBM recently halved the price of TM/2 and appears interested in marketing it; in any case, the system is used extensively internally for IBM's own multilingual documentation production

A version of TM/2 for Windows is currently in preparation. Apparently, this will not include the MT API.

Vendor

IBM Deutschland
PS Box Hanns Klemmstr. 45
D-71003 Böblingen
GERMANY
Tel: +49 7031 164400
Fax: +49 7031 164609

G.0.19 LinguaTech

Products

MTX, MTX Reference

Comment

Formerly known as Mercury (in the US) and Termex (in Europe), MTX (DOS) is the oldest and most widely used terminology management package (DOS). In 1992, LinguaTech introduced MTX Reference, a scaled down version for users who do not need to compile termbases, but only refer to them.

The MTX engine has also been licensed by such publishers as Collins and Larousse for use as retrieval software in their electronic offerings. As a result, the MTX format has become a de facto standard in electronic publishing, although it has never been promoted as such.

MTX is developed by LinguaTech (Provo, Utah) and distributed in Europe by Eurolux. Eurolux also offers termbases in MTX format for various domains and language pairs; these materials have been acquired from a variety of third parties.

Vendor

Eurolux Computers (European distributors)
11, rue de Wormeldange
Gonderange, 6180
LUXEMBOURG
Tel: +352 789 443
Fax: +352 789 820

G.0.20 Microtac

Product

Language Assistant

Comment

The Language Assistant series (DOS, Windows) are integrated packages aimed primarily at people writing in foreign languages. They contain bilingual dictionaries licenced from Random House, conjugation generators, and a bidirectional batch translation mode for sentences. Over two-hundred thousand copies have been sold at US$ 79.00.

Spanish, French, German, and Italian versions of Language Assistant are available

Vendor

Microtac Software
9375 Jutland Drive, Suite 110
San Diego, CA 92117
USA
Tel: +1 619 272 5700
Fax: +1 619 272 9734

G.0.21 Linguistic Systems

Product

Euroglot

Comment

Euroglot is an innovative multilingual thesaurus (DOS) with a concept rather than lexeme orientation. It has conjugation and user-dictionary functions. It is available for six European languages.

Euroglot is primarily aimed at people producing correspondence in foreign languages.

Vendor

Linguistic Systems BV
Postbus 1186
Nijmegen, 6501 BD
THE NETHERLANDS
Tel: +31 80 226302
Fax: +31 80 242116

G.0.22 SITE/Eurolang

Product

Optimizer

Comment

Introduced in March, 1994, Eurolang Optimizer (Windows, Unix) is a high-end integrated translation tool based on client/server architecture. It is designed to be used in conjuction with Word for Windows and Framemaker, and has extensive strong terminology and translation memory management facilities.

Vendor

SITE
12 rue de Reims / BP 35
F-94702 Maison Alfort Cedex
FRANCE
Tel: +33 1 45 13 05 00
Fax: +33 1 45 13 05 59

G.0.23 Soft-Linc

Product

Trans-Link

Comment

A multilingual thesaurus (Windows, Macintosh) similar in design and scope to Euroglot (see Linguistic Systems).

Vendor

Soft-Linc NV
Mechelsesteenweg 148
B-2650 Antwerp
BELGIUM
Tel: +32 3 454 0 454
Fax: +32 3 454 0 436

G.0.24 STAR

Product

Transit

Comment

Transit (DOS, Windows) is an integrated translator's workstation package.

Vendor

STAR AG
Postfach 4
CH-8260 Stein am Rhein
SWITZERLAND
Tel: +41 54 41 22 11
Fax: +41 54 41 22 55

G.0.25 TA Electronic Publishing

Products

TransDic, OfficeDic

Comment

TA Elec Publishing (formerly part of Triumph Adler) was established in 1993 and has introduced a range of inexpensive translation tools and bilingual dictionaries (Windows). Initially, offerings were for English, German, and French language pairs. TA hopes to establish its retrieval software as a standard platform for electronic publishing.

Vendor

TA Electronic Publishing
Führter Stra

G.0.26 e

212
D-90429 Nürnberg 80
GERMANY
Tel: +49 911 32 42 910
Fax: +49 911 32 42 919

G.0.27 Telesoft

Product

Deja Vu

Comment

An inexpensive translation memory tool (Windows) which can be used in conjunction with word-processing packages. It includes a function for aligning sentences in existing translations.

Released at the end of 1993, Deja Vu is jointly marketed by a Spanish company called Telesoft and an American company called TranSoft.

Vendor

Telesoft
El Alcornocal BL5, 2
E-28410 Manzanares
SPAIN
Tel: +34 1 853 0627
Fax: +34 1 853 0627

G.0.28 TRADOS

Products

MultiTerm, MultiTerm Dictionary, Translator's Workbench II, TAlign

Comment

MultiTerm is a terminology management package (DOS, Windows). Translator's Workbench II (DOS, Windows) is an integrated version of MultiTerm and a translation memory module. The DOS version requires its own editor; the Windows version works in conjunction with other wordprocessing packages. MultiTerm Dictionary is a dictionary compiler. TAlign is an alignment tool for creating bilingual translation memory files from existing translations.

The Trados packages have superseded the INK TextTools, which are still supported but no longer actively marketed by its developer, now known as Donnelley Language Solutions.

Vendor

Trados GmbH
Gutenbergstra

G.0.29 e

4 b
D-70176 Stuttgart
GERMANY
Tel: +49 711 62 70 68/9
Fax: +49 711 616 7 45

G.0.30 Vertaal

Product

Vertaal!

Comment

Vertaal! is a series of bilingual electronic dictionaries (DOS, Windows) designed as simple aids for writing in a foreign language. The retrieval software was developed by Dutch softwarehouse AND Software; the dictionaries are those of Dutch publisher Spectrum.

Language pairs available: Dutch-English, Dutch-French, Dutch-German, and Dutch-Spanish

Vendor

Vertaal Publishers BV
Postbus 2740
3562 LC Utrecht
THE NETHERLANDS
Tel: +31 30-650650
Fax: +31 30-60850

Appendix H

Feature Checklist Examples

This appendix contains an example feature checklist, made specifically in the context of the EAGLES work. It deals only with translation memory. Its major methodological view is that tests should be rigorous and efficient.

Similar feature checklists should be developed for other software serving translators.

H.1 Feature checklist for translation memory

The first thing to do in the construction of a feature checklist is the definition of the types of software (programs or components thereof), or, in terms of section 2.2, the types of members of the set O that one is concerned with. This is done in section E.3.1.

The second step is then to define useful attributes for each type of object of evaluation. This is what is done below. Attributes for a feature checklist are given for each type. As the main user-oriented organizing principle, we used the 6 quality characteristics for software defined in ISO-9126 (see section 2.1.1). Moreover, the checklist given here is constructed on the principle of rigorous test methods: attributes are included only if they take well-defined formal values and a reliable test procedure (inspection or benchmarking) is imaginable.

H.1.1 TM updating/maintenance

H.1.1.1 Alignment

1 Is the segmentation (of SL-text and its translation) executed automatically? (yes/no)
- if so:
  - units of segmentation by selection? (yes/no)
    - if so: How many and what kind of units?
    - if not: What kind of standard unit?
  - are there possibilities to check/correct the output (of the segmentation procedure)? (yes/no)
  - what is the average speed of the procedure?
- if not: see above (same questions)

2 Is the alignment procedure executed automatically? (yes/no)
- if so:
  - are there user-defined regions? (yes/no)
    - if so: How many and what kind of regions?
    - if not: What kind of standard region?
  - are there possibilities to check/correct the output (of the alignment procedure)? (yes/no)
  - what is the average speed of the procedure?
- if not: see above (same questions)

3 What is the average percentage of correct alignments?
4 Does the complex function (segmentation followed by alignment) impose conditions/restrictions on the input text as to:
- character sets? (yes/no)
- format? (yes/no)
- markup? (yes/no)
- other restrictions? (yes/no)
- if so: which/number/seriousness? (scale)

5 Do the following features appear unchanged in the outputs (of the segmentation as well as the alignment procedure):
- format? (yes/no)
- markup? (yes/no)

6 What is the minimum number of primitive actions (macros included) for performing alignment (including automatic segmentation) on two given texts? (on the assumption that any conditions have been satisfied beforehand)

H.1.1.2 Importing an aligned SL- and TL-segment to the translation memory database

1 Is it possible to import the aligned text pair directly, ie without having to perform other tasks
- an existing TM database? (yes/no)
- a 'new' (empty) TM database? (yes/no)
- several TM databases at a time? (yes/no)

2 Does the import function impose conditions/restrictions on the input text as to:
- format? (yes/no)
- markup? (yes/no)
- other restrictions? (yes/no)
- if so: which/number/seriousness? (scale)

3 Do the following features appear unchanged in the output:
- markup (of the added text pair)? (yes/no)

4 How does the program react if an aligned SL- and TL-segment are imported and one of these segments has already been stored in the TM database in question:
- the new segment is added to the TM database
- the new segment isn't added to the TM database
- the old segment is deleted from the TM database
- a warning appears: the user has to make a choice?

5 What is the minimum number of primitive actions (macros included) for importing an aligned SL- and TL-segment to a TM database? (on the assumption that any conditions have been satisfied beforehand)

H.1.1.3 Adding a SL-segment and its translation to TM while translating in TM mode

1 Is it possible to have the SL-segment and its translation added to TM database automatically ('automatic updating')?
if so:
- is it possible to select another TM database to add the sentence to (indicate another database as the active one)?
- is it possible to deactivate the automatic updating function in individual cases?

2 How does the program react if a SL-segment and its translation are added to the TM database and one of these segments has already been stored in that database:
- the new segment is added to the TM database
- the new segment isn't added to the TM database

- the old segment is deleted from the TM database
- a warning appears: the user has to make a choice?

3 What is the minimum number of primitive actions (macros included) for adding a SL- and TL-segment to the TM database? (on the assumption that any conditions have been satisfied beforehand)

H.1.1.4 Modifying existing contents of TM's (apart from adding/importing)

1 Is it possible to modify a translation unit (a SL-sentence and its translation(s)) that has already been stored in a TM database? (yes/no)
if so:
- is it possible to modify the SL-segment? (yes/no)
- is it possible to modify the TL-segment? (yes/no)

2 Is it possible to delete an entire translation unit from a TM database? (yes/no)

3 Is it possible to export a TM database? (yes/no)
if so:
- do the following features appear unchanged in the output:
  - format? (yes/no)
  - markup? (yes/no)
- is it possible to obtain a subset of translation units from an existing TM database using selection criteria? (yes/no)
  if so:
  - How many and what kind of criteria?
  - Is it possible to use combinations of criteria (e.g. negation, conjunction, disjunction)

4 Is it possible to transfer a translation unit directly (no separate export/import activities) from one TM database to another? (yes/no)

5 Is it possible to merge two or more TM databases into one treating the entire database as one unit? (yes/no)

6 Is it possible to combine two TM databases that have one language in common resulting in one database with a new SL-TL combination? (yes/no)

7 In case of exporting/transferring a translation unit to another TM database or merging two TM databases: how does the program react if the SL-segment would appear twice in the 'new' TM database:
- the new segment is added to the TM database
- the new segment isn't added to the TM database
- the old segment is deleted from the TM database
- a warning appears: the user has to make a choice?

8 Is it possible to invert the contents of a TM database (the source language is regarded as the target language and the target language as the source language)? (yes/no)

9 What is the minimum number of primitive actions (macros included (like deleting an entire translation unit with a hotkey)) for performing the different tasks described above?

H.1.2 TM application

H.1.2.1 The productivity of the translation memory

1 Define benchmarks to determine the productivity of the translation memory mode translating a certain text type (ratio number of hits and size of TM database(s), exact/fuzzy matches)

2 Does the package contain a utility for previous analysis of the average match value of the text to be translated and one or more TM databases?

H.1.2.2 Translating a SL-text in TM mode and having TM generate/propose a match per SL-segment

1 Does the program impose conditions/restrictions on the input text (the SL-text which has to be imported into the editor) as to:
- characterset? (yes/no) (if so: ... (to be elaborated))
- format? (yes/no) (if so: ... (to be elaborated))
- markup? (yes/no) (if so: ... (to be elaborated))
- other restrictions? (yes/no)

2 Possibility to apply in interactive or batch mode?

3 Possibility to check/correct the output (i.e. the translation(s) suggested by the utility):
- is it possible to expand or reduce the segment (indicated by the program) to be translated (max. and min. length)? (yes/no)
- is the match value presented on the screen? (yes/no)
- is it clear how the percentage similarity is calculated (i.e. is the matching SL-segment presented together with the SL-segment to be translated, are the differences/correspondences between the SL-segments indicated, ...)? (yes/no)
- are 'standard' editing functions available to the translator to modify the suggested translation? (yes/no)
- is it possible to modify the suggested translation while having present on the screen both SL-segments? (yes/no)

4 Definition of a match (one or more sentences suggested by the program, which have a certain similarity to the SL-segment to be translated):
- do markup features/formatting tags constitute a distinctive variable in the procedure of finding a match? (yes/no)
- do punctuation marks constitute a distinctive variable in the procedure of finding a match? (yes/no)
- does upper/lower case constitute a distinctive variable in the procedure of finding a match? (yes/no)
- how much alike must two SL-segments be to be regarded by the program as an exact (i.e. 100%) match (implying the three preceding items and the possibility that the SL-segment to be translated and a stored SL-segment only differ as to one (or more) numerical performance datum (or data))? (scale)
- is the system able to deal with so-called fuzzy matches? (i.e. the SL-segment to be translated has only a certain similarity to one or more stored SL-segments) (yes/no)
  if so:
  - is it possible (for the translator) to set a boundary value for the fuzzy match percentages (i.e. a minimum fuzzy match value)? (yes/no)
  - is there a fuzzy match value that can be considered a useful minimum value in that it is the lowest percentage to generate (on the average) acceptable matching segments ? (yes/no)
  - is the length of the segment to be translated (and analysed by TM) restricted to a min. and max. number of words? (depends on the method used to compare the SL-segment to stored SL-segments) (yes/no)

5 Are the following features present which benefit a correct output:
- utility for preserving the markup of the SL-segment in the translation (situations: tags in stored SL-segment/no tags in SL-segment to be translated or the other way round; both segments contain tags)? (yes/no)
- utility for relating/comparing one complex SL-segment in the text to two (or more) component SL-segments in the TM database or the other way round? (yes/no)
- utility for suggesting more than one SL-segment from the TM database in case of a fuzzy match with identical match values (or maybe different cases of an exact match), or suggesting more than one translation which is stored in TM with one and the same SL-segment? (yes/no)
- utility (filters) for selecting translations beforehand: certain subject area, client, etc.? (yes/no)

6 What is the minimum number of primitive actions (macros included) for performing the different tasks involved in translating a SL-text in TM mode? (also: is automatic translation of segments with a 100% match possible?)

H.1.3 The translator's workbench program

Here we look at the integration of TM within the entire package.

H.1.3.1 Openness

With what other functions can the TM function interact:

- editors/word processors (list)
- DTP/programs for printing texts (list)
- database (text retrieval) tools (list)

H.1.3.2 Editing

1 Does the program contain a 'translation specific' editor, i.e. include special functions relevant for the translation task? (yes/no)
   - if so: - How many and what kind of functions? (list)

2 Special treatment of tags:
   - are tags protected from being overwritten during the translation process? (yes/no)
   - are tags hidden to prevent extensive screen clutter? (yes/no)
   - other utilities? (yes/no)

3 Is an interface for layout and printing included in the editing functions?

H.1.3.3 Using the termbank program

Looking up words/terms in association with the translation memory:

- is it possible to look up a word/term with a hotkey (while working in translation memory mode)?
- is it possible to transfer the translation of the word/term to the text using a cut-and-paste facility?
- are 'known' words/terms indicated (e.g. highlighted) in the source text?
- is it possible to call up the terminology entry of a term with a hotkey (assuming that the entire entry isn't provided automatically when looking up a word/term)?
- is it possible to define and then select attributes which can be used as filters to search selectively?
- Is it possible to have known terms translated automatically after a fuzzy match?

Appendix I

A Practical Evaluation of Writers' Aids

I.1 Introduction

This appendix gives an indication of how some grammar checkers can be evaluated in the light of EAGLES work on writers' aids. The tests were not intended to be exhaustive, but to validate that the methodology chosen is on the right track. In addition to a grammar checker for English, a grammar checker for another language was also tested, to verify that the methodology is applicable regardless of the specific language in question.

Two grammar checkers were taken as a basis for testing: one dealing with the English language and one with French. Since we are not dealing with the evaluation of a specific product, but indeed with the testing of our own methodology, we prefer to refer to the grammar checkers without using their real commercial names. So we refer to the English grammar checker as *E1*, while the French one will be referred to as *F1*.

For the first round of testing undertaken on *E1* and *F1*, two types of texts have been used. One was extracted from the collection of texts of economic bulletins written by Union Bank of Switzerland (UBS). These are intended for the general public, and although some economic terms inevitably appear, they do not feature too specialised a vocabulary. These texts were supplied in their revised, published format. An advantage of using them was that they appear both in English and French, so that it has been possible to compare the performance of the two grammar checkers on equivalent texts. We have reasons to believe that the transfer of the texts to us resulted in occasional typographical errors. This was the case for most hyphenated words, where the first letter of the second word disappeared, as for example, *Etats-Unis* vs. *Etatsnis* in the French texts, and *long-term* vs. *longerm* in the English ones.

The other type of text involved a few series of sentences written especially to test the checkers. These can in turn be subdivided into sentences that contain a grammatical error, sentences that contain no errors and sentences that contain no errors but feature a certain structure meant to trap the software.

For the second round of testing only constructed material was used.

The overall impression is that both checkers do catch various types of error, but that overflagging (that is, indicating as an error a sentence or a part of speech that is in fact correct) is fairly high, which greatly reduces the usefulness of the systems.

As the testing for *E1* and *F1* was conducted on two types of text, the outcome of the testing activity will be described separately for each test type.

I.1.1 Testing on the UBS corpora

UBS publishes an economic bulletin roughly once a month, tracing economic and financial trends for the Swiss and international markets. It contains section titles and figures, percentages, dates, names of the different stock markets and indices, etc. In general, the language used is not specialised jargon and the style used is of a fairly high standard.

The main advantage in using this kind of text is that the evaluator can see what kinds of real-life errors can sneak in and, conversely, where real-life false flagging is likely to happen. The texts we had available are the ones given to the printer for final printing; we can believe that they have gone through revision and approval by the appropriate services. However, on several occasions the checkers' flagging indicated a real error, which may be seen as proof of the usefulness of grammar checkers in general.

One of the options that the checkers offer is that the text can be corrected in batch mode and the corrected file subsequently stored on the hard disk. This is very handy in general, and especially for the kind of testing intended, so that the final result from the grammar checker can be evaluated as a whole and not just as unconnected output from the screen.

False flagging is probably the most serious problem with grammar checkers. It suggests that a sentence is incorrect or inappropriate, or that the style is too familiar or too pedantic. In the long run, users can get so put off by a checker's false flagging that they will decide to do without it. Clearly, this aspect is as important as spotting the error, if not more.

As previously said, there are not many error examples in the UBS texts used, given that they were texts intended for final printing. This should be kept in mind in order to keep things in perspective with regard to the amount of false flagging the checker does. In a sense, if we wanted to use the number of false flaggings versus the number of correct flaggings as a measure, these texts could not be used as testing material, since the result would be unfairly biased.

It should also be noted that not all flagging messages are presented with the same level of explicitness. Indeed, there are different types of message relating to errors. Quite often, the checkers would draw the user's attention to some possible error with a vague message like: *Verify this sentence: there is probably a syntactic error*; it is then up to the user to figure out what the problem might be. When the program finds more precise indications that something is not grammatical, the error message is something like: *Verify if there should be agreement between* `noun` *and* `adjective`*:* `noun` *is singular and* `adjective` *is plural.* [1] On subject–verb agreement, the typical error message is: *If* `noun` *is the subject of* `verb`*, then there is wrong agreement.* Especially regarding the French checker, it is only on very short and simple sentences that the error is presented as a statement, and not just as a possibility.

False friends and homonyms are also a great source of overflagging: since many words can be mistakenly used for one another, the checker prompts the user every time it comes across a potential lexical misuse and points out that such a word can be mistaken for another. Usually, it also gives very brief descriptions of the meaning of the two words in question. [2] This feature is not a source of false flagging per se: it draws attention to a potential word misuse without control over the flagging. However, there is a menu option to deactivate flagging for both false friends and homonyms, which puts the user in control of the flagging.

[1] This particular example is related to the testing of French.

[2] For instance, in the French checker: <##'CONJONCTURE': ETAT, CIRCONSTANCES. NE CONFONDEZ PAS AVEC 'CONJECTURE': SUPPOSITION, HYPOTHESE.#> *'conjuncture': state, circumstances. Do not confuse with conjecture: supposition, hypothesis*; in the English checker: |–'AFFECT' AS A VERB MEANS 'INFLUENCE'. YOU MAY WANT THE VERB 'EFFECT', MEANING 'BRING ABOUT' OR 'CAUSE'.–|.

I.1.1.1 Grammar checker for English

The grammar checker for English, *E1*, provides the possibility of running the program in both interactive and batch mode, where in the latter case the text is marked with the errors detected. In addition, there is also the option of marking one problem at a time during interactive proofreading. These two options, however, present some weaknesses. First, the information provided by marking a problem is usually less rich than that provided in interactive mode. In fact, whereas the former only advises there is an error, the latter also provides suggestions for replacement. This is especially the case (but not the only one) when spelling errors occur. Secondly, in marking a particular problem during an interactive session, the program inserts a piece of text between `|--` and `--|` before the error occurrence. Surprisingly, after inserting such a text, the program keeps on proofreading the document considering the text inserted as part of the sentence to be tested, thus detecting errors such as subject–verb agreement or incomplete sentences.

I.1.1.1.1 Well spotted errors As previously noted, the texts used had already been proofread, therefore they do not present many actual errors. However, the texts contain some errors, e.g. in punctuation (such as extra space before commas), capitalisation (a sentence starting with an uncapitalised word or presence of a comma after *e.g.* or *i.e.*) and subject–verb agreement (even in not straightforward cases), and most of these are well caught by the checker. In addition, errors in expressions involving restrictions on uncountable or countable nouns are usually detected.

I.1.1.1.2 False flagging False flaggings found by a grammar checker may be classified according to the level at which the flagging occurs: paragraph, sentence, phrase or word level.

Paragraph level: Errors at this level are more concerned with the style than with grammar, e.g. a paragraph consisting of one sentence only. This flagging occurs in more or less the same way as for the French version. More details are to be found in section I.1.1.2.

Sentence level: In addition to false flagging related to style such as use of long sentences, use of passive voice or use of too many prepositional phrases, quite a lot of false errors are detected because of the ambiguity of many English words that can be identified both as a noun and as a verb. In many of these cases the checker detects errors like subject–verb agreement, or incomplete sentence, or gives warnings about some verbs that do not require objects. Since *E1* can view the parts of speech assigned to each word (only in case of error detection) of a sentence, it has been possible to understand where it failed. An example of such false flagging is the following sentence:

> `|--`*(THIS DOESN'T SEEM TO BE A COMPLETE SENTENCE.)*`--|` *A comparison of the ECU market rate with the reference or theoretical interest rate shows that the actual three-month ECU deposit rate averaged 0.24 of a percentage point above the theoretical ECU interest rate between September 1984 and June 1989.*

In this sentence the word *shows* is considered a plural noun instead of a verb. However, sometimes, even if no word is recognised as a verb by the grammar checker, no error is flagged. [3]

Another noteworthy false error detected by *E1* is concerned with parenthesis. The grammar checker does not consider the possibility that the text included between parentheses can be

[3] It is not always the case that in such words priority is given to nouns rather than verbs; in fact, in the sequence *interest rates*, which appears many times throughout the texts, *rates* is almost always recognised as a verb, the subject of which is *interest*.

an independent sentence and therefore looks for agreement. [4] In addition, we can mention the following example to show another relevant drawback of the grammar checker:

> |--*(THIS DOESN'T SEEM TO BE A COMPLETE SENTENCE.)*--| *Accounting for 7.1% of the EC's GNP (1988; in ECU) and over 4.4% in EC internal trade (average of share in exports and imports), the peseta will have a weighting of 5.3% in the ECU currency basket.*

E1 considers a sentence as a string ending with a character from the set {., :, ;}. The first warning shown is relative to the sentence

Accounting for 7.1% of the EC's GNP (1988)

which is incomplete. Furthermore, when it checks the second sentence, it loses the knowledge about the occurrence of the first parenthesis, and therefore warns about a mismatched parenthesis.

Phrase level: *E1* fails to recognise expressions such as *up to now* (suggesting the use of *too* instead of *to* if the meaning is 'also' or 'excessively'), or *in turn* (warning that an article or another modifier usually precedes the word *turn*). In addition, whenever the phrase *half of* occurs, it suggests the use of only *half*, even if the whole phrase is *the first half of 1989* or *second half of the month.*

Word level: As mentioned before, some noise occurred when transferring the file containing the texts and most words containing hyphenation have been affected (e.g. *shorterm* vs. *short-term*, *threeonth* vs. *three-month*, *2ear* vs. *2-year*). This implied a great deal of involuntary testing of spelling errors, most of which have been detected, with the exception of *2ear* or analogous mistakes. For the spelling errors due to this noise, the only correct suggestion for replacement provided by the checker was *three-month.* Two other types of false flaggings usually occurred: (1) all *-ing* forms used as nouns are not accepted in the plural forms; (2) it is suggested that a hyphen is not used with prefixes such as *non- inter-*, but if the hyphen is cancelled, a spelling error occurs.

I.1.1.2 Grammar checker for French

The grammar checker for French that we tested will be referred to as *F1.* Although this checker offers the option of running the program in batch mode and storing the corrected file on hard disk, it has been impossible to save an output file with more than about 2,000 words, due possibly to the Windows or the DOS version that our PC runs, but this blocks the proper functioning of *F1.* This is a major drawback for the user and we had to divide the original files to enable the system to process and save the corrected output. (As an indication, the two original files used were 6103 and 5831 words long).

I.1.1.2.1 Well spotted errors On the texts used, *F1* correctly detected punctuation mistakes such as mis-matched parentheses, or an extra space between a word and a comma or full stop. This might be viewed as secondary, but it is one of the errors most overlooked by human revision.

Amongst the traditional grammar errors, *F1* concentrates on subject–verb agreement, adjective–noun agreement and spelling errors. On the spelling front, if a wrong spelling is detected in a verbal form, the suggestion list can propose the full declension of the verb, along with other suggestions from dictionary look-up. This is an appreciable facility that should be acknowledged.

[4] For instance: *On June 16, 1989, the EC Finance Ministers tentatively set the new weightings for the currencies in the ECU basket for the* |--*(THE SINGULAR SUBJECT 'PERIOD' TAKES A SINGULAR VERB, NOT THE PLURAL VERB 'SEE'.)*--| *period from September 1989 to September 1994 (see Table 1, last column).*

F1 also behaves fairly well in spotting wrong agreement in verbal phrases with a past participle form. This is indeed one of those rules that most commonly confuses people and triggers errors in the agreement. In the texts used, this did not happen very often, only a couple of times, and in constructions where it is very easy to get confused.

As an example, *F1* correctly spotted the following error in subject–verb agreement, where, in fact, we have a cross-reference between two sentences. Note that the checker makes assumptions as to which NP is likely to be the subject: in the first case there is indeed an error of agreement, but not in the second one.

> *Le renoncement à une majoration des taux en Europe et au Japon l' < # SI 'RENONCEMENT' EST LE SUJET DE 'ONT', IL Y A UNE FAUTE D'ACCORD. # > ont [le dollar] < # SI LE SYNTAGME COORDONNE EST LE SUJET DE 'A', IL Y A UNE FAUTE D'ACCORD. # > au contraire stimulé et la nette compression du déficit commercial en juillet lui a donné des ailes.*[5]

In this sentence, the subject of the first verb *ont* (present, 3rd person plural of *être*, 'to be') is indeed the singular noun *renoncement* ('renouncement') and not *taux* ('rates'); however, the second part of the sentence does not contain an error, and the checker seems to consider the conjunction to be between two NPs (but then which ones?) and not between two sentences. (In the English version the conjunction was rendered as two separate sentences separated by a full stop.)

Amongst style errors, *F1* flagged those sentences starting with a digit and suggests writing out the relevant number in full or re-organising the sentence.

I.1.1.2.2 False flagging As previously done for the description of the English version, the false flaggings found by the checker for French will be grouped according to the level at which the flagging occurs: paragraph, sentence, phrase and word level.

Paragraph level: Fairly often, *F1* would complain of the "appearance" of a paragraph, pointing out that "usually, a paragraph has more than one sentence".[6] Very often, this type of false flagging appears when the checker encounters the title of a section (especially if it contains something the checker may interpret as an inflected verb form). In such cases, the length of the sentence is no more than one line. The checker would flag any paragraph (regardless of its length) if it contained only one sentence.

Sentence level: There are two major types of false flagging in this category — either the sentence is found to be too long (which is a simple word count: the number of words used as a basis may be set by the user) or the verb form is not found, is not of the desired voice (passive forms are always flagged as bad style) or is the wrong mood (where subjunctive or conjunctive moods must be used, as part of a construction with a conjunction or a subordinate clause). False flagging is triggered in those cases where, for instance, both subjunctive and indicative are correct, or when the checker wrongly assumes that a conjuction controls a given verb.

Phrase level: This is where the most false flagging appears. *F1* links parts of speech that do not in fact form a phrase, requiring agreement where it is wrong to do so. (For example, *ce sont … avant tout les agents économiques qui …* ("it is … above all the economic agents who …") where agreement between *tout* ('all') — singular — and *agents* — plural — is suggested). Another well-known source of grammatical problems for French is the agreement

[5] The translation, taken form the corresponding English version of the same bulletin, is: *The nonmaterialization of interest rate increases in Europe and Japan further fueled its [the dollar's] momentum. A sharp upsurge followed in the wake of the news that the U. S. trade gap had narrowed significantly in July.*

[6] We agree that this is borderline false flagging, since *E1* does not mark it as an error per se, but merely gives a suggestion of the typical style.

of past participles. The program fails to recognise some correct constructions involving these. The phrase *la décision du le 18 février* ("the decision [taken on] February 18th"), was flagged as wrong, on the grounds that *18* was plural, whereas *février* was singular.

Word level: *F1* has a built-in spelling checker that checks the text before checking for grammatical errors. However, some words are flagged by the grammar checker as inappropriate or bad style (especially some conjunctions). Particularly, the checker forces the user to abbreviate *Fr.* (the usual abbreviation for Swiss franc) as *F* or *FF*, meaning French franc (as opposed to Swiss or Belgian, etc.). This type of error is not caught by the spelling checker (where it would be easy to just add the acronym to the personalised dictionary), but by the grammar checker, which makes it impossible to add it. It also does not allow numbers to be written as figures, even when intended as prices, etc.

I.1.2 Testing the ad hoc sentences

A few sentences were especially created to test the checker, with the aim of checking the coverage of the grammar and ascertaining whether the program would false flag some intentional traps. Focus has been on NPs, maintaining a certain equivalence between the English and French testing, even if some changes had to be made to account for differences between the two languages.

The list is not intended to be exhaustive, but can definitely be a starting point for an evaluation of grammar checkers.

I.1.2.1 Grammar checker for English

The testing of ad hoc sentences has been concentrated on a specific class of Noun Phrase errors, i.e. determiner–noun disagreement (see section D.5). Some initial testing has also been performed in relation to phonetic adjacency agreement and the genitive.

It has been possible to make some comparisons between different versions of the same program. Many erroneous results found in the older version of the grammar checker were no longer present in the newer one, e.g. concerning mass nouns.

As proposed in section D.7.4, a set of correct inputs, a set of erroneous inputs and a set of misleading ones have been considered. These inputs are defined such that the same construction may appear both in simple realisations and in more complex ones. In some cases this has turned out to be very useful. For example, let us consider the noun phrase:

All this work.

The input has been considered correct. But the input:

All this good work

was considered wrong. A third combination:

All this interesting work

was correct.

Most of the erroneous results given by the program were due to confusion between nouns and verbs, even in the presence of verbs. Usually, correct sentences were recognised as such, but, in many incorrect sentences, the error reported by the grammar checker did not match the supposed one.

Another weakness presented in *E1* concerns ordering problems. The following erroneous sentences were in fact considered correct:

**Many these painters.*

**Some fresh other milk.*

I.1.2.2 Grammar checker for French

In general, *F1* could spot straightforward errors of agreement (noun–article or noun–adjective subject–verb agreement and agreement of past participles). In some cases, a complex subject NP (e.g. *the kittens near the door which is shut are ...*) would make the checker interpret the wrong noun as the subject, flagging for wrong agreement.

The checker is capable of spotting typographical errors, e.g. mis-matched brackets, spaces before a full-stop, non-inclusion of a question mark within inverted quotation marks, etc.

The test sentences focus on a few key points. Some are aimed at the agreement of *tout* ('all'), which can be either an adjective or an adverb, and will consequently change its behaviour in agreement. Testing of the list of NPs showed that straightforward errors would be caught, but *F1* would overlook errors in a slightly more complicated construction.

The test sentences were kept fairly simple and short. After having used *F1* on the UBS texts, there was no need to check it further for side effects due to long and convoluted sentences.

One of the tests was to see if the program would recognise non-existent verb forms: in French, some verbs are never used in some tenses, which is difficult to keep in mind when writing. *F1* failed to spot that the verb *parfaire* ('to perfect, put the finishing touches to') is only used in compound tenses or in the infinitive, and thus did not flag the sentence:

**Albert parfaisait la statue que Robert avait laissée incomplète*
("Albert was giving the finishing touch to the statue that Robert had left incomplete").

In another case, for the verb *renaître* ('to be reborn'), rarely used in compound tenses, the checker points out, erroneously, that the user should have written *avoir* ('to have') as the auxiliary.

Some sentences were aimed at testing, somewhat more exhaustively, agreement for words that can be both an adjective and an adverb. *Tout* ('all') is among those words, and a few sentences were written to represent most of the possible constructions in which *tout* can appear. Here, the checker seems to base its recognition of *tout* as an adjective or adverb simply on the basis of the word's proximity to a noun or a verb. For instance, in the sentence

**Ils sont tout partis*
("They have all left")

where *tout* refers to the pronoun and should therefore be in the plural form *tous*, the checker does not spot the error. Conversely, it marks as wrong the correct sentence

Ils se sonts tous brûlé les mains
("They all got their hands burned")

pointing out that *tout* used as an adverb should not be in the plural form. Since these kinds of error are fairly frequent even for French mother tongue speakers, the grammar checker does not help as much as it could and in fact confuses the user even further.

As a side effect of testing for adjective-adverb usage, the following false flagging was detected. The sentence

Ils sont tous partis
("They have all left")

was tested on *F1*. The checker correctly assumes that *tous* is an adjective referring to the subject, but also assumes incorrectly that *partis* is not the main verb (past participle of *partir* ('to leave')), but the noun *parti* (meaning '(political) party'. This is evident because it prompts the user with a message warning for possible misuse of a word, in this case as a false friend or homonym. (The message reads more or less: *Do not confuse "party": union of interests, with "part": element of a whole*).

Another common source of error is homophones. These are difficult to catch, since most of the time the words concerned share the same syntactic category and it is only at the semantic level that they differ. However, *tout* is pronounced like *toux* ('cough'). In a sentence like

**l'enfant était au lit avec la tout*
(intended:
l'enfant était au lit avec la toux
'the child was in bed with a cough')

the checker should have been able to spot the error. *Tout* can be used as a noun, in which case its gender is masculine, but in the sentence above the article is feminine.

When working on the list of NPs, *F1* could not correctly recognise recognition of vowels/consonants that guide the choice of determiners. As in English, where there is a difference in using *a/an* according (grossly) to the first letter of the following word, in French some determiners have a double form and the choice of which to use is dependent on the following phonetic environment. For instance, *ce* ('this') should be written *cet* when followed by a vowel or an unpronounced *h*. The checker flagged correctly

**ce homme*
("this man")

but

**ce énorme travail*
("this enormous work")

was allowed to pass un-noticed.

However, *F1* was accurate in spotting the correct usage of the article *le/l'* or *la/l'*, where the vowel of the article is dropped and the apostrophe appears before a word beginning with a vowel or an unpronounced *h*.

I.2 Results of tests on NPs

This section shows the results of the first testing round. In the following tables *ok* means that the result is the correct one; *?* means that an unexpected error is returned; and *X* means that the grammar checker does not respond to the error.

I.2.1 Tests on English NPs

Input	Results	Notes
A painter.	ok	
A painter and a sculptor.	ok	
The painter.	ok	
This painter.	ok	
That painter.	ok	
These painters.	ok	
Those painters.	ok	
Some painters.	ok	
Some painter.	ok	
Many painters.	ok	
*Many painter.	ok	
*Many dog.	X,?	dog as a verb
*Much painter.	ok	
Another painter.	ok	
*Another painters.	ok	
Some other dog.	ok	
Some other dogs.	ok	
*Some another dog.	X	
*Some another dogs.	X,?	dog as a verb
*Some another painter.	X	
*Some many painters.	ok	
These many painters.	ok	
These many other painters.	ok	
*This much painter.	ok	
*Many these painters.	X	
*More painter.	ok	
Few painters.	ok	
Most painters.	ok	
Any painter.	ok	
Any painters.	ok	
No painters.	ok	
No young painters.	ok	
No very young painters.	ok	
*Both painter.	ok	
Both the painter and the sculptor.	ok	
Both painters.	ok	
Both young painters.	ok	
Both young and old painters.	ok	
*Both many and few painters.	X	
*Both all and no painters.	X	
*Every painters.	ok	

Input	Results	Notes
*All painter.	ok	
All painters.	ok	
Almost all painters.	ok	
Almost all these painters.	ok	
Fewer painters.	ok	
*Fewer painter.	ok	
*Less painter.	ok	
*Less painters.	ok	
Some milk.	ok	
*Many milk.	X	
*No milks.	X,?	milk as a verb
*Every milk.	ok	
*Every sand.	ok	
Some baggage.	ok	
*Some baggages.	ok	spelling error
*One sand.	ok	
*Both milk.	X	
*First sand.	X	
*Few air.	X	
Less air.	ok	
*Less airs.	X	
*Fewer air.	X	
Fewer airs.	ok	
Some fresh milk.	ok	
Some very fresh milk.	ok	
Some other fresh milk.	?	milk as a verb
Some other fresh milk comes from Holland.	ok	
*Some fresh other milk comes from Holland.	X	
*Many fresh milk comes from Holland.	X	
All this work.	ok	
All these works.	ok	
All this big work.	?	adverb (bigly)
All this interesting work.	ok	
All these big works.	ok	
*All this big works.	X	

Input	Results	Notes
A paper.	ok	
*An paper.	ok	
*A umbrella.	ok	
An umbrella.	ok	
A unique paper.	ok	
*An unique paper.	ok	
A beautiful umbrella.	ok	
*An U.N. employee.	ok	
A European citizen.	ok	
*An European citizen.	ok	
An UNCED.	X	
*A X-ray.	X	
An X-ray.	ok	
A hotel.	ok	
*An hotel.	ok	
The engineer's book.	ok	
The engineers' book.	ok	
*The engineers's book.	ok	
*The engineers book.	X	
*The book is her's.	X,?	spelling error
The book is hers.	ok	
Mary's book.	ok	
*Mary book.	X	
*Mary book is on the table.	X	
The company owner's name is Jack.	ok	

I.2.2 Tests on French NPs

This section shows the results of the first testing round. In the following tables *ok* means that the result is the correct one; *?* means that an unexpected error is returned; and *X* means that the grammar checker does not respond to the error.

Input	Results	Notes
Un peintre	ok	
Le peintre.	ok	
Un peintre et un sculpteur.	ok	
Ce peintre.	ok	
Ces peintres	ok	
* Quelques peintre	ok	
* Beaucoup peintres.	X	
* Beaucoup de peintre.	ok	
Beaucoup de peintres.	ok	
Un autre peintre.	ok	
Des autres peintres.	ok	
D'autres peintres.	ok	
Ces nombreux autres peintres.	ok	
Ces nombreux peintres.	ok	
Le hall.	ok	
Les halles.	ok	
Les hôtels.	ok	
* L'hôtels.	ok	
* L'hall de l'hôtel	ok	
Ces nombreux peintres.	ok	
Ces nombreux autres peintres.	ok	
Peu de peintres.	ok	
Peu de peintre.	ok	
Aucun peintre.	ok	
Aucun jeune peintre.	ok	
Aucun très jeune peintre.	ok	
* Ce beaucoup de peintres.	X	
* Plus de peintre.	X	
* Peu peintre.	X	
* Aucun peintres.	ok	
* Les deux peintre.	ok	
Les deux peintres.	ok	
Les deux jeunes peintres.	ok	
* Tous les peintre.	X	
Tous les peintres.	ok	
Tous les jeunes peintres.	ok	
Tous les jeunes et les vieux peintres.	ok	
* Chaque peintres.	ok	
* Chaques peintres.	ok	
* Moins de peintre.	X	
Chaque peintre.	ok	
Presque chaque peintre.	ok	
Presque tous les peintres.	ok	
Presque tous ces peintres.	ok	
Moins de peintres.	ok	

Input	Results	Notes
Du lait.	ok	
Beaucoup de lait.	ok	
* Bien de lait.	X	
* Chaque lait.	X	
* Tous les laits.	X	
Du lait frais.	ok	
Du lait très frais.	ok	
Des fruits frais.	ok	
D'autres fruits frais.	ok	
* D'autre fruits frais.	X	
Des bagages	ok	
* Du bagage.	X	
Tout ce travail.	ok	
Tous ces travaux.	ok	
Tout ce grand travail.	ok	
Tout cet énorme travail.	ok	
* Tout ces énormes travaux.	ok	
* Tout ce énorme travail.	X	
* Tout ces énormes travail.	ok	
* Tout ces travaux.	ok	
* Un autoroute.	ok	
Une autoroute.	ok	
Une acné.	ok	
* Ce homme.	ok	
* Ce énorme travail.	X	
* Ce animal.	ok	
* Cet fleur.	ok	
L'ONU.	ok	
Un employé de l'ONU.	ok	
L'OMS.	ok	

I.3 Second round of testing

I.3.1 Introduction

In the second round of testing, the framework as developed in chapter 2 and in more detail in Appendix D was applied in an evaluation of the English grammar checker, *E1*. An evaluation was carried out of the functionality of the checker as stated in the accompanying documentation. We then applied the results of the evaluation to see whether the checker could meet the requirements of a specific user group. This comparison led to an extra round of testing of some of the specific user requirements which were not mentioned in the documentation.

The aim of performing the evaluation was to verify that the framework is indeed suited for the evaluation of an NLP product.

I.3.2 Functionality requirements

In the evaluation described here the first two functionality requirements have been employed. They are repeated here (cf. section D.5):

1. The system should specify the grammatical errors made by the writer in such a way that the end-user is able to correct them.

2. The system should not signal an error where no such error is present.

In fact, the second part of requirement 1 has not been taken into account, i.e. the actual quality of the advice has not been evaluated.

I.3.3 Measure

The measures used to evaluate the checker are the measures of precision and recall. These measures indicate:

- how often is grammatical incorrectness rejected (precision of a grammar checker)
- how often is grammatical correctness accepted (recall of a grammar checker)

Precision measures the percentage of grammatical errors correctly detected by the grammar checker. Recall measures the percentage of the grammatically correct items accepted by the grammar checker. At the same time, the recall percentage may be used to calculate the percentage of false flaggings, i.e. grammatically correct items rejected by the grammar checker: The percentage of false flaggings is obtained by subtracting the recall percentage from 100.

The two measures complement each other and are both necessary in the evaluation of a grammar checker. To know whether the checker detects the errors that we test for without knowing whether it in fact finds errors in all the input sentences is not useful. Similarly, to know whether it accepts all the correct input sentences without knowing whether it accepts all the input sentences, incorrect as well as correct ones, is useless.

I.3.4 Method

Again we are adhering to the guidelines from the Framework Model (section D.7). The methods suggested are concerned with amount and composition of test material. In these evaluations we have not been so much concerned with amount as with composition, but in reality a fairly large set of error examples are called for. We have concentrated on the composition, and followed the guidelines repeated here:.

- For each attribute there is a horizontal tripartition: a set of erroneous inputs, a set of correct inputs and possibly a set of deliberately misleading inputs (setting 'traps' for the system);
- For each attribute there is a vertical ordering: the inputs cover the range from the most simple instantiation of a construction to the more complex realisations of it.

Measuring precision and recall is done by running a grammar checker on the test material and counting the number of rejected and accepted items, respectively.

I.3.5 Evaluation of the grammar checker wrt stated requirements

Here the set of errors on which the evaluation should be based is the set of errors that the system states it detects. These errors are listed here:

- placement of adjective errors
- confusion between adjective and adverb errors

- article errors
- clause errors (subject/verb group, dependent clauses)
- comma splices
- comparative/superlative use
- conjunction errors
- double negatives
- homonym errors
- incomplete sentences
- incorrect verb forms
- infinitive errors
- noun phrase errors
- object of verb errors
- preposition errors
- pronoun case errors
- pronoun number disagreement
- relative pronoun errors
- run-on sentences
- subject/verb disagreement
- subordination errors
- tense shifts

To cover the total amount of grammatical problems mentioned above would be very time consuming, and as the usefulness of our framework, rather than the actual performance of the grammar checker, is our main concern, only some of the problems are covered in this study.

It should also be mentioned that we have not had access to already existing test suites for this testing. Consequently, the test material we have used is far from being sufficient for a *real* evaluation. So again, the testing below is meant to show the feasibility of the methodology, not to be an example of a full scale testing.

In the testing, we have used the methodology outlined above, i.e. provided sentences in pairs, a wrong one and a correct one.

I.3.5.1 Results

Adjectives/adverbs: *E1* did not perform very well on errors of adverbs and adjective usage, and the distinction between the use of these two parts of speech.

For example there are certain adjectives that can only function as either premodifiers or postmodifiers but not both. *E1* did not react to any of these. Below are two examples of sentences with an adjective that can only be premodifying or postmodifying, respectively.

* *The fool is utter.*

**The afraid man.*

As the methodology indicates, the phenomena were also checked with the corresponding correct sentences below. These were also considered correct by the grammar checker:

He is an utter fool.

The man is afraid.

The checker also accepted adjectives which were incorrectly used as adverbs, so for example the following ungrammatical sentence was accepted:

**He spoke brief and frank.*

The checker accepted ill-formed comparatives, such as the following:

**The house is beautifuller than I thought.*

**The mouse is more below the table.*

In the two examples above we actually have two problems of ill-formed comparatives, (i) a problem of comparative of an adjective that can have an comparative form, but with *more* not by inflection, and (ii) there is an attempt to form a comparative of a preposition which is not possible.

- precision: 17,6%
- recall: 100%

Articles: *E1* produced some surprising results on missing articles.

The checker sometimes caught a sentence with a missing article, sometimes it did not. Thus, the first of the following examples was not caught by the checker but the second was.

**He was brave man.*

**He was good boy.*

- precision: 62,5%
- recall: 83,3%

Clauses: In general *E1* handles clauses satisfactorily, with some exceptions where it falsely flags correct sentences as errors for example in some adverbial constructions such as;

As was natural, he married her.

E1 flags this as an error. The error message states that 'was' seems to be lacking a subject.

E1 also falsely flags some appositive clauses as incorrect, e.g.

A message that he would be late arrived by special delivery.

In this case the error message is that the sentence does not seem to be complete.

- precision: 15,4%
- recall: 69,2%

Prepositions: The correct choice of prepositions is a well-known problem for second language writers of English — or any other second language for that matter. Prepositions also posed a problem for the grammar checker.

Most of the tested sentences with wrong prepositions were not flagged by the checker. This was all the more surprising since most of the errors were constructed around the fact that certain verbs subcategorise for only one or a few prepositions. So for example,

**The package consisted with many things.*

was not flagged as an error although *consist* subcategorises for the prepositions *in* or *of*. Similarly, the following sentence was also accepted by the checker,

**He pays attention on it.*

although the preposition in this phrasal verb should always be *to*.

However, some of the errors are not possible for the grammar checker to correct without semantic knowledge, and these should, therefore, not be considered to be errors in this connection, e.g.

**He laughed with my words.*

is semantically ill-formed but this is not due to any subcategorisation restrictions since *laugh* does not subcategorise for a prepositional phrase complement. So the following sentence

He laughed with the others.

is perfectly acceptable.

- precision: 35,7%
- recall: 92,9%

Subject/verb disagreement: The problems relating to concord are primarily subject-verb concord in sentences starting with *there*, uncountable nouns, indefinite pronouns, or where the verb is divided from the subject noun. The checker catches some of these errors but not all of them as can be seen in the following examples:

**The police has never found his murderer.*

This sentence was caught by *E1* (*police* is a plural noun). The next two sentences, however, were not.

**There are violence in the school.*

**Some maintains that this is correct.*

- precision: 66,7%
- recall: 100%

I.3.5.2 Comments

As can be seen, it is maybe time consuming, but otherwise fairly easy to test a grammar checker according to its specifications.

I.3.6 Evaluation of the grammar checker wrt user requirements

Above, we tested some of the quality characteristics which are part of the coverage claimed by the producers in their documentation. However, as mentioned earlier, an evaluation has to take its point of departure in actual user requirements. As it is a big task in itself to establish such user requirements, we chose to use the list provided by the producer and to supplement it by requirements of a specific user group. The only indication of intended users of *E1* is a small paragraph in the documentation stating that professionals as well as students may benefit from using the grammar checker. We have chosen to focus on second language students.

I.3.6.1 User group and error types

The second language students whose requirements we describe, are Danish students aged 16–19 years having been taught English in school for 7–10 years.

Below are the ten most frequent error types for the user group according to Færch, Haastrup and Phillipson (1984). The list is not ordered, but the error types mentioned cover about 80% of the total amount of errors, so together they constitute the most important part.

1. concord (subject/verb concord, usually in sentences starting with 'there', uncountable nouns, indefinite pronouns, or the verb is divided from the subject noun)
2. articles (usually an overuse of articles)
3. predicate (very heterogenous group, often direct transfer from Danish or verbal complementation errors)
4. verb phrase (errors in tense, aspect, auxiliary, modal verbs, conjugation errors)
5. tense shifts
6. noun phrase (errors of nominalization of adjectives or numerals)
7. word order (e.g. placement of adverbs)
8. adverbial phrase (structure problems)
9. aspect (mostly progressive aspect problems)
10. number (no marking of distributive plural, conjugation of uncountables)

Some of these problems are specific to Danish writers, some specific to second language writers of English in general, and some belong to the more general area of errors of writers of English. As could be hoped, some of these error types are already part of the list established by the grammar checker provider. But there are also types which were not explicitly mentioned. Below we have chosen to test two of these, i.e. word order and aspect.

I.3.6.2 Results

Word order: Word order errors for the user group concern the correct placement of adverbs, for example

**Violence is today a big problem.*

In this sentence *today* is not placed correctly. However, the grammar checker did not flag this error.

- precision: 50%
- recall: 100%

Aspect: Since progressive aspect does not exist in Danish, its use can pose a problem for the user group who has a tendency to overuse it. The behaviour of the checker is somewhat unpredictable here. Whilst the first two examples below were caught by the checker the last two were not.

**I am knowing the language.*

**He is owning the car.*

**I am understanding that she is coming.*

**The book is belonging to her.*

- precision: 15,4%
- recall: 100%

I.3.7 Conclusion

We have shown how a grammar checker may be tested, relying partly on its own specifications, partly on the requirements of a specific user group.

The test results mentioned above are based on so small amounts of data that no general conclusion can be drawn from them. However, a few general comments can be made: some grammatical phenomena are not handled very well, whereas others pose no problems. False flagging can be a problem on the sentence level, and no flagging as regards prepositions is an overwhelming problem — at least if this is a common source of error of the type of user whose requirements are taken into account. The most important problem, however, is the lack of consistency. Sentences containing the same problem are not treated the same way as in the examples under Articles and Aspect above. However, the semantic problems we encountered cannot be ascribed to the grammar checker, as they are outside its scope.

To sum up, we have chosen the following strategy for testing a grammar checker:

- Establish quality characteristics on the basis of the functionality stated in the documentation of the grammar checker.
- Create a test suite corresponding to the quality characteristics and following the principles mentioned above under Method.
- Specify quality characteristics on the basis of a requirements analysis of a particular user group.
- Create a test suite, preferably building on real-life errors (plus the corresponding correct sentences).
- Use the precison/recall measures.

I.3.8 Evaluation of the framework

We have seen that the framework model can easily be used for the evaluation of grammar checkers. It is possible to establish measurable quality characteristics corresponding to user requirements. The measures of precision and recall seem well suited for this type of evaluation.

Appendix J

Evaluation of Knowledge Management Systems

J.1 Introduction

The purpose of EAGLES evaluation work on knowledge management systems[1] (KMSs) is twofold:

- To set up a potential scenario for the use of *information retrieval* (IR) systems in combination with natural language processing (NLP) technology, either for supporting NLP (application-oriented view) or for improving IR systems with respect to their (computational) linguistic competence (engineering-oriented view);
- To define criteria for the evaluation of KMSs.

Initially, we aimed at concentrating on general information retrieval, possibly with a natural language front end, with emphasis on different kinds of semi-automatic (computer-assisted) and automatic *indexing* for *text (document) retrieval*. However, due to lack of manpower resources and after having scanned the market for IR products that can, at least to a certain extent, support NLP applications, such as the analysis of corpora or the construction of term banks, we have decided to focus on one specific area of application, namely *technical documentation and document production*. In the near future, this area will be essential for massive industrial and academic research and development, e.g. for product localisation operations, value-added services and multilingual language technologies, in particular when taking into account the amalgamation of computers, media and telecommunications.

The focus of work will be on technical documentation, because general texts would be too imprecise due to the fact that such texts can be interpreted very differently depending on the intellectual interests, background and knowledge of a potential user of an IR system; technical texts, however, are usually not subject to broad interpretation. Therefore, we see an IR system not only as a general information provider (broad view) for a specific subject, but also as a knowledge-based support tool for the extraction and maintenance of the heterogeneous knowledge needed for technical documentation (application-oriented view).

The result of our market survey was disappointing in that commercially available systems provide only very limited functionality with respect to their applicability to NLP. Although IR provides good solutions for storing textual data, in contrast to general database management systems, it does not allow for complex structuring of the data and flexible access methods, because of its strict character string orientation.

[1] *EAGLES only NB*: We have changed the title of our section in order to avoid confusion with the Information Management System (IMS) of IBM.

Today, several industrial companies have sophisticated tools available for supporting their in-house information retrieval procedures and their technical documentation. However, these products are not yet available to the public and therefore they cannot be taken into account for EAGLES purposes.

Given this situation, we will not evaluate existing systems (or only to a limited extent), as is done in other sections of this chapter, but will rather *define application oriented requirements* for a KMS that is able to support *large-scale NLP related applications*. Based on these requirements we will be able to specify (to some degree) the *functionalities* of such a KMS, which will also lead to *basic evaluation methods* (perhaps a kind of de-facto standard). These methods will take into account the methods applied in other sections, i.e. *adequacy*, *diagnosis* and *progress evaluation*. However, relevant research and development systems in this area are also taken into account, such as the CODE system (Skuce and Lethbridge, 1993), RELATIO/IR (Rahmstorf, 1994), SYSTEM QUIRK (Ahmad, 1993) of the ESPRIT project Translator's Workbench (TWB) and The Information Theater (XeroxParc, 1992).

As already mentioned, the area to be studied is *technical documentation* and the focus will be on an *information (knowledge) repository* that supports a technical writer during the various processes of documentation and manual production, such as planning, information (knowledge) acquisition, drafting, reviewing and production.

Given the limited time and budget frame of EAGLES, this work will be mainly based on work-in-progress and the research interests of the author. However, we will keep the results as open to generalisation as possible.

The impact of the work will be twofold. On the one hand, it will provide developers with guidelines for the design of future knowledge-based IR systems, thus promoting the creation of competition which may improve products, and, on the other hand, it will allow potential users to evaluate new products according to their needs.

This section is organised as follows: first, in section J.2 we will set up the general KMS scenario which will form the basis for the specification of a knowledge-based assistant for technical writers which is briefly presented in section J.3. The focal point of this system will be its knowledge management component that will facilitate the building of the system's knowledge base. We will describe the general requirements for such a system and give an overview of its architecture. The requirements are mainly derived from empirical research results which are available as a separate document(Ross, in press, forthcoming) . This then leads us to different evaluation criteria outlined in section J.4. The criteria will be initially defined here and will thus form the basis of future work within the EAGLES.

J.2 The KMS scenario

J.2.1 Knowledge management systems

A *knowledge management system* (KMS) is the software framework (toolbox) that is intended to assist, via knowledge processing functions, those who desire to formulate and retrieve knowledge for different applications, such as system design and specification, term bank construction, documentation or ontology design for (multilingual) language processing. The various tools of such a framework should help users to originate and organise ideas or understand and communicate ideas more easily and accurately than can be done with most current tools. A KMS is an integrated multifunctional system that can support all main knowledge management and knowledge processing activities, such as:

- Capturing;

- Organising;
- Classifying and understanding;
- Debugging and editing;
- Finding and retrieving;
- Disseminating, transferring and sharing knowledge.

Current knowledge management systems, in particular those in the field of information retrieval, are

- Too narrow in many respects. For example, one application, one type of user, one type of knowledge representation, one type of knowledge operation, etc.;
- Too hard to use. For example, specialised knowledge is needed and long training curves are necessary;
- Not widely known or available.

One main task of a KMS is to search for specific information. This is mainly done by an information retrieval component which will be central for our evaluation work.

J.2.2 Information retrieval

According to ISO 2382/1 (1984) *information retrieval* is defined as:

> *...[the] actions, methods and procedures for recovering stored data to provide information on a given subject.*

Under *actions*, the ISO document subsumes text indexing, inquiry analysis and relevance analysis; with *data*, it identifies text, tables, diagrams, speech, video, etc., and also hypermedia in order to distinguish between non-linear structured texts or parts of texts, and linear texts (documents); with *information*, it associates the relevant knowledge that is needed for supporting problem solving, knowledge acquisition, etc.; and with *subject*, it associates a concept, as opposed to a character string (word).

The central problem of IR is the analysis and measurement of the *relevance* of the stored information, i.e. the relation between requested information and retrieved information. In other words, given an information inquiry, the IR system has to check whether the stored information is relevant to the inquiry. Traditionally, this problem has been solved by organising the database that is used for the search as an *inverted file* of the *significant character strings* occurring in stored texts, i.e. the inverted file specifies where the strings occur in the texts. Normally, the strings used are natural language words excluding determiners, conjunctions, prepositions and the like (which are known as *stop-words*). An inquiry to an IR system is then composed of these character strings combined by *Boolean operators* and some specific additional features such as *contextual* or *positional operators*. Since there is no linguistic analysis of the semantics of the stored texts or of the inquiries, IR systems are mostly *domain independent*.

Today, there are several approaches to improving IR systems in general, particularly approaches oriented towards *relevance analysis*. Among these are, for example, dialogue components (menu-based or natural language based) and statistical approaches. The former are mainly used to improve the use of a given query language according to pre-defined user profiles (novice, expert, etc.), whereas the latter will improve the relevance analysis directly. There is also on-going work in the field of introducing NLP technology into IR systems (cf. for example Salton, Buckley and Smith (1990)) which obviously would be a further improvement of IR systems. In addition, future IR systems will have to take into account new styles of publishing and electronic communication, including capabilities investigated in the field of *virtual reality* (cf. Sherman and Judkins (1993)).

Other problematic areas in IR which are closely related to the relevance problem involve the *measures* of *precision*, *recall* and *specificity*. These may be glossed as follows (see section J.2.4 for further details):

Precision: A text presented as an answer to an inquiry should contain only relevant information.

Recall: All texts containing relevant information should be found and presented as an answer to an inquiry.

Specificity: Information requested (by the user) or offered by the texts stored in the database should be expressible without limitations. It should be possible to specify any subject and any detail of the requested information the system has to search for.

J.2.3 Search units

In information retrieval, we distinguish three kinds of search units:

1. **Individual documents:** the focus of the search is on a specific document and not on a list of possible documents.
2. **Any document:** the focus is on any document containing a specific character string defined in a search inquiry.
3. **Information on a specific subject:** the focus is on documents detailing the given subject (concept).

For each kind of search we can identify typical user applications. In case 1, the user knows that there is one single document which contains the key information he is looking for. In case 2, the user wants to study (analyse) the contexts of character strings, but not the information which may be communicated by the retrieved documents; typical users are terminologists and linguists. In case 3, the user does not want a unique document: the information he wants must belong to the selected subject about which he hopes to improve his knowledge.

Thus, the application areas for IR are very diverse. We may distinguish between the following search methods, based on the kind of search:

Reference text search (bibliographic IR): Besides standard data and fact retrieval, this would include specialised applications, such as IR for specific storage media (e.g. compact disks) or mixed media (e.g. image retrieval), and IR combined with problem solving (e.g. expert systems).

Non-reference text search (full text systems): This would include IR for on-line product information, reports, journals, etc., and for short lived information (e.g. electronic mail) that could be stored in personal databases or in a company's archives.

Specialised text search (texts generated for electronic use and support): This would include IR for encyclopaedic information (short technical texts) and for hypertext (non-linear structured text).

J.2.4 Precision, recall and specificity

The standard measures for IR are *recall* and *precision*. Assuming that:

- RET is the set of all texts the system has retrieved for a specific inquiry;
- REL is the set of relevant texts for a specific inquiry;
- RETREL is the set of the retrieved relevant texts, i.e. RETREL = RET $\cap$ REL.

then precision and recall measures are obtained as follows:

precision = # RETREL / # RET
recall = # RETREL / # REL

The basic information unit presented to a user is a *search report* that contains a partial document text which could be either a linear paragraph or a textual unit when using a hypertext system.

Specificity is concerned with the *information* or *query language* of an IR system. Often such a language is:

1. Not correctly applied by the user;
2. Not expressive enough for a specific search inquiry.

The first problem is mainly based on the abstractness and complexity of a query language regarding different system users, whereas the second problem is rooted in the complexity of natural language expressions which are part of an inquiry. For example, an inquiry looking for an explanation of *stand-by orbital communication system* has to identify that *stand-by* modifies *system*, in contrast to an inquiry about *stand-by power supply system* where *stand-by* modifies *power supply* instead of *system*. This becomes more crucial when the inquiries are embedded in Boolean expressions.

J.2.5 Application and measurement criteria

The criteria for the application of IR systems are manifold; among them, the most relevant are:

- Reference or primary information (bibliographic information *vs.* complete text);
- Form of information (text *vs.* facts, data, etc.);
- Combination of different media (text *vs.* multimedia);
- Life-time (e-mail *vs.* documentation);
- Copyrights and licences (own information *vs.* foreign information);
- Encapsulation of information (unrestricted access *vs.* controlled access, e.g. CD);
- Text length (short texts *vs.* long texts, e.g. product manuals);
- Text form (text for print media *vs.* text for electronic media);
- Linearity (linear text *vs.* hypertext).

These criteria apply to documentation and document production in several ways, as we will show in section J.3, and are therefore also criteria that have to be taken into account when evaluating systems in this field.

Measures that are essential for IR users can be summarised as:

- Investigation amount (time and costs);
- Procurement amount (time and costs);
- Evaluation amount (time, redundancy of information).

These open a further dimension for the evaluation procedure, concentrating on user-related measurements in contrast to system-centred measurements, such as speed, hardware and software requirements, etc.

J.2.6 Summary

Current IR approaches are based on character string oriented techniques and inverted files. At present, there is no full integration of computational linguistic techniques, apart from morphological approaches or shallow syntactic analyses (cf.̃below). Their typical functionality accounts for:

- Unstructured text and structured records;
- Inquiries based on Boolean logic;
- Separation to the right or left of a word;
- Simple distance operators (adjacent, in sentence or phrase, in paragraph).

The obvious advantages are application independence and near language independence (so-called *cross-products*). However, precision, recall and specificity have to be improved. The limitations of systems are mainly based on the expressivity of the *information language* (cf.ãbove).

Current trends in IR can be characterised by the techniques used:

1. Statistics-based;
2. Menu- or dialogue-based with relevance feedback and user models;
3. Natural language based, e.g. thesauri, synonym techniques and generation of inflected forms;
4. Knowledge-based.

The first three techniques offer well-established formalisms which allow for rich expressiveness and reliable deduction mechanisms. However, regarding the fourth technique, at present knowledge representation is associated with individual system design. This is because procedural construction of a knowledge base (KB) is not permitted. Thus, one finds that idiosyncratic interpretations of the predicates and other constructs of the KB naturally occur. In addition, KBs have a high amount of documentation effort and are limited to small worlds. This latter limitation is a crucial restriction on the extensibility of a KB as well as on the possibility of changing application domain. In the following section, we will present potential solutions to solve some of these shortcomings.

J.3 Case study: KBAS-TW

In the following, we will set up an architecture for a knowledge management system in the field of technical documentation and document production. The design of such a system will also trigger the identification and selection of evaluation criteria and methods in the field and, thus, may lead to the specification of a general evaluation methodology for knowledge management systems.

We start with a general description of the situation in the field of documentation in order to identify the potential software tools that may support the different processes in this area. Then we will define a knowledge-based assistant for technical writers based on the sorts of knowledge that are involved in the primary identified processes. The main focus will be on the design and the construction of the knowledge base of the assistant system.

J.3.1 General overview

Today, more technical documentation has to be produced and translated in a shorter time because of the massive increase in technological developments and innovations. The knowledge about a product in the form of complete and reliable information (general information, user manual, reference manual, etc.) has to come with the product, otherwise the product will have have no real chance on the market. From a consumer's point of view, a bad product manual counts as a serious shortcoming of the product. Product documentation and user manuals often do not fulfill the basic requirements of user-oriented presentation and information reliability. This is due to professional, organisational and technological weaknesses in the production process of the documentation. Better results in the production of this type of knowledge are required. Technical documentation becomes a better knowledge and information product if more know-how is invested in its production.

Currently, no tool assists the technical writer or the translator[2] in a coherent way during all document development stages and, in particular, at the level of a document's domain of discourse. To some extent, recent developments in the field of terminological knowledge bases may support the technical author in this respect (cf. Meyer, Bowker and Eck (1992) and Schütz and Ripplinger (1993)), however, functionalities of content related operations still remain poor. One estimate of current tools in the field of desktop publishing is that they are effective for only 20% of consumer instructions and 8% of manuals (cf.for example Hofmann and Heino (1992)). Today, the computer-assisted organisation of technical documentation and translation as an integral part of the documentation process is still in its infancy. There is a need for tools that permit the technical writer to acquire, store, up-date, edit, etc., product and document related information and knowledge. Thus, a better tool is needed: a workbench for technical writers which integrates different tools into a documentation environment.

We have to make a distinction between technical writing and general text editing. Professional writing is quite rich in its methodological knowledge; by the mere effect of routine and professional competence, professional writers dispose of more ready-made methods, rules and patterns than non-professionals. The task-related knowledge of a technical writer may be formalisable, but often is only implicit. This means that, in order to design and specify a workbench for technical documentation, an analysis of the tasks of a technical writer has to be done; as far as we know, such an analysis has not been done until now (cf.below).

We have to bear in mind that different kinds of knowledge are involved in these tasks. Knowledge-based systems (KBSs) are powerful means to manage heterogeneous types of knowledge. Normally, a KBS supports and solves a task in a domain where knowledge can be represented in a compiled expression that is efficient for the task. In our scenario, we have the problem of separating domain knowledge from knowledge about various tasks. However, these types of knowledge are heavily interrelated. For example, acquisition, explanation and validation tasks can be seen as general

[2] The envisaged user of the proposed system is either a technical writer or a translator.

tasks, but they rely on domain-specific knowledge which may then trigger the tasks in specific ways. Thus, the user should have the possibility to tailor a knowledge base to his specific needs.

Certainly, there are difficulties for such a KBS to reach an industrial stage. Problems occur because, in practice, substitutions are similar but not identical. It is necessary to configure automatically a KBS to have clones of the initial KBS. These systems act like a knowledge configurator that is used by the user himself to tailor the KBS to his specific needs. In such a case the user is an expert. The user and the knowledge based assistant system (KBAS) may profit from each other. The man-machine interaction for acquiring and refining knowledge extends the role of expert systems to *expert supporting systems*. On top of all this, knowledge is volatile and evolves as projects evolve. Thus, the KBAS will not be a 'single-use' system. It will be used for maintenance and updating in an endless process of knowledge acquisition.

In a company, knowledge is accumulated over many years and it may not be available in any form directly tractable by computational means. Thus, the recovery of this knowledge is a task of extreme importance. It is becoming increasingly urgent to share knowledge sources among applications that can use them in different ways. A natural approach considers that the central repository would be managed with various tools, each applicable to and manipulating some part of the knowledge and information stored within the repository.

J.3.2 A KBAS for technical writers

We centred our investigations around the different tasks for technical documentation, focussing on the activities and requirements of professional writers and translators of technical manuals. Preliminary work involved the study of available commercial instruction sets. Then we interviewed and observed technical writers, translators and decision makers during their work. While this research focus is sufficiently narrow to address the fundamental issues in detail, it revealed information that certainly enhances and triggers the design and specification of computational tools for supporting the production process of technical documents (cf. Ross (1993)), as well as evaluation methodologies. The results of the data and task analyses of our study led to the building of an information repository, which is considered a powerful knowledge base, for the knowledge that is necessary to support the documentation process and the generation of technical manuals for different readerships. It is integrated in an overall object-oriented architectural environment, the knowledge-based assistant system KBAS-TW, which will permit the technical writer to tailor knowledge to his personal needs.

The development of such a knowledge base for technical documentation relies on cooperation with the user. In our approach, the user, either being an experienced technical writer or a knowledge engineer, will be considered an expert. Both user and system will follow the same goal, i.e. the realisation of the knowledge base, but their tasks are complementary.

At the beginning, we suppose that various sources of information are available to the KBAS-TW and the user. Such chunks of information are all those documents which have already been written on a given object of the real world (old version of the manual which should be produced, technical specifications, technical sheets, market analysis reports, customer specifications, own notes, teaching sheets, etc.) by different persons who supply the technical writer with information (the customer, the developer, the marketing manager, etc.) during the life cycle of the object. At this first stage, the goal of the cooperation between the user and the KBAS-TW is to:

- Acquire the various types of knowledge and of documents from the information sources;
- Formalise and control a homogeneous repository that will contain all the pieces of information and knowledge deemed necessary for documentation;
- Negotiate with the user the tailoring of the knowledge.

The end of this stage will not be the end of the cooperation. Indeed, objects of the real world evolve and the cooperation will be followed by the acquisition of new information from the user, who will then be the unique source of information.

Additionally, the system has to possess a *model of the tasks* to be undertaken by the user. Thus, there should be one component that controls all communication flows with the user, the information sources and the knowledge base. In this scenario, the user is supposed to consult only the knowledge base. In this way, we aim to ensure high quality of the work done during the cooperation with a minimum of communication overheads.

The development of the functional specifications of KBAS-TW has followed the typical path to building an expert system (cf. for example Slage and Wick (1988) and Gruber (1989)), namely to:

- Get the technical writer motivated;
- Get an idea of the problems the technical writer is confronted with;
- Reconstruct the problem solving process of the technical writer in a principled way;
- Extract the relevant information for this problem solving process;
- Implement and test the knowledge base against this information.

J.3.3 Designing and specifying the knowledge base

J.3.3.1 Design methodology

The first phase in the design of the knowledge base is a general knowledge acquisition phase in which we will follow to some degree the guidelines specified for knowledge-based systems as described, for example, in Boose and Gaines (1990). The goal is to:

- Obtain the specific knowledge that will enable the system to perform its task;
- Define how the system can support the technical writer in his various tasks;
- Specify how the system could be implemented in a working environment for technical writers and translators. This includes:
  - An evaluation of the tools that are necessary for the working environment,
  - An analysis of the communication flows between the technical writer and the persons who supply him with information or the resources that provide him with information.

In our work, this phase was divided into two stages: an initial inquiry stage and a detailed investigation stage, including an analysis of technical documents (of a specific domain). In the second stage, we interviewed and observed 12 technical writers, translators and decision makers at different phases of the production process of documentation and technical manuals during a 12 month period according to the methods described in Norman (1985) and Card, Moran and Newell (1983) (cf. Ross (1991) and Ross and Verdret (1992)). These interviews were first open and then semi-structured; the answers and the observations were reported in a protocol book. For the formalisation of the identified tasks, we used a modified question-answering procedure (Graesser, 1978).

The formal model that guided our empirical research is that of Flower and Hayes (1980), which provides a general framework for the writing process. To it, we have added the knowledge acquisition phase of the technical writer for preparing the documentation process, because we observed that the technical writer draws upon a plan or outline to divide the writing task into nearly independent subtasks, chosen so that they can be achieved separately, without constant attention to possible interactions between them. This plan acts as a *structural constraint* on the delineation and ordering of subtasks. The subtasks are not fully independent; instead they are subject to *contextual constraints*, such as convention of writing styles, simplified natural language (syntax and semantics), abbreviation consistency or terminology. Another set of constraints is characterised as *resource constraints* — the text may need to include particular references, figures and graphics; or one has to avoid certain abbreviations or jargon. A detailed analysis of the production process is described in Ross (1993); it consists of 8 main phases:

1. Planning;
2. Acquisition;
3. Organisation;
4. Drafting;
5. Reviewing;
6. Production;
7. Printing;
8. Archiving.

Our focus is on phases 2–4 and 6 which constitute the actual writing process. In most cases, phase 1 is also executed by a technical writer, but decision makers are also involved in this process. Phase 5 is to be seen as a control phase of the product, according to the planning phase and other constraints of the enterprise. The last two phases are performed by persons other than the technical writer.

Our belief is that a true purpose-driven information repository can only be approached asymptotically. Our construction method involves a stepwise folding in of different information clusters one at a time. We have to distinguish between:

- Knowledge about the domain (world) under consideration;
- Knowledge about the textual objects of that domain and their internal structuring;
- Knowledge about the various tasks involved during the process of document production, including different knowledge acquisition procedures for maintaining these tasks;
- Knowledge about the actual linguistic realisations of domain entities in different languages, especially in terms of domain specific terminology.

We believe that appropriate interfaces can be designed and implemented with systems that deal with knowledge-based computational terminology, natural language processing and machine translation.

J.3.3.2 Knowledge acquisition

We have to distinguish between the knowledge acquisition phase that builds up a first instantiation of the information repository and the knowledge acquisition phase that tailors the information repository to a specific need. The former is normally done by a knowledge engineer, whereas the latter is done by the technical writer and corresponds to the phase in which the technical writer gathers all the information he needs in order to write a technical document (phase 2 above). This phase has two main tasks:

1. Retrieval of potential *concepts* from existing knowledge sources;
2. Audience analysis.

The first task involves the identification of specific relations among groups of concepts, the clustering of concepts and the modification of their representation for the envisaged tasks, and results in:

- Instantiated individual concepts;
- Application oriented clusters of concepts;
- A (rough) skeleton of the document.

In this phase, one has to obey several constraining features, such as norms that, on the one hand, are specified by the customer and that, on the other hand, are inherited by the domain, applicable tools and the sources which supply information. In order to perform this task effectively, the technical writer needs appropriate support in the form of a sophisticated information retrieval tool and tools that allow for consistent treatment of terminology and norms.

The second main task of the technical writer is *audience analysis*. This task is crucial because technical writing is essentially a *message planning* task. The quality of the technical document depends on how well the audience analysis has been done. The main goals are to:

- Identify the major purposes the document will serve;
- Clarify the rhetorical register;
- Identify and rank potential readerships.

The output of this task is:

- Clarification of the rhetorical goal;
- Choice of the appropriate terminology;
- A priority list of readers;
- A listing of the desired actions of the envisaged audience.

In practice, technical writers perform audience analysis intuitively. Therefore, there is a need for a methodology on how to adapt the knowledge to different audiences supported by appropriate software tools.

The knowledge acquisition phase for initialising the repository is concerned with the conceptual analysis of the domain, on the one hand, and on the other hand, with the building up of the actual conceptual structure of the domain, and the conceptual engineering of the tasks that are relevant for documentation and manual production. For the former, we have envisaged re-using already existing domain ontologies. By doing this, we shall also prove the feasibility of sharing knowledge resources across projects, which is essential to real world applications. If the ontology contains links to linguistic realisations and formalised concept definitions, it is possible to define the relationships between documentation models and associated tasks, and the ontologies. This, then, allows for a task oriented view over the ontology.

The knowledge obtained in both acquisition phases is then processed in three steps for our purposes:

1. Building for each information cluster a taxonomy of conceptual realisations required to describe that information cluster;
2. Organising the domain entities in terms of the different conceptual structures;
3. Merging the result with existing ontologies in a well-defined way in order to enable the link to production oriented organisations.

A further step would be to merge the results with existing interlingual ontologies in order to enable the link to language-dependent linguistic realisations (multilinguality).

The methodology adopted is based not on intuitive grounds about what is and is not 'true' about the considered world, but instead on empirical and practical concerns, namely what data, information and knowledge the document production processes require in order to perform different tasks.

After the initial knowledge acquisition process, the resulting knowledge structures contain, declaratively and explicitly represented, those distinctions required to control the document production process in various ways and the acquisition of new information in a guided and tailorable way. The latter is a crucial aspect in KBSs, especially in real world industrial applications. Thus, the knowledge base maintains three different conceptual structures, i.e. a domain model, a document model and a task model, to permit the reusability in other application domains. The links between the models specify the application-oriented relationships between the knowledge sources.

For the actual representational devices of the repository, we are considering, besides classical knowledge representation formalisms, a formalism based on typed feature logics (cf. Carpenter (1992)) which will allow for an appropriate (virtual) interface to unification-based natural language processing systems.

J.3.3.3 Knowledge organisation

In the knowledge organisation phase, information from the identified knowledge sources is layered into appropriate description devices. The goal of the technical writer is to transform the outline from the knowledge acquisition phase into a coherent representation by obeying the same constraints as in the acquisition phase. This involves such processes as:

- Analysis;
- Synthesis;
- Building an abstract structure;
- Refining the structure.

The output is normally a hierarchy of concepts which correspond to the model of the document and which is formalisable in terms of a *document planning language* (DPL).

J.3.3.4 Document production and reviewing

The document production phase consists mainly of writing and reviewing tasks. This phase is realised by linguistic and graphical means according to the kind of document and its envisaged layout. As an intermediate product, the technical writer constructs a *block of technical text*. He linearises sequences of characters divided into words, lines, sections and paragraphs, etc. He reviews the global organisation of the document, moves and refits large units of text. Then, he focusses on coherence relations among smaller segments within an intermediate-scale frame of reference, such as sections and paragraphs.

The output of this phase is a coherent technical document. The process corresponds to a linguistic encoding of the document plan that was the output of the knowledge organisation phase. The main constraint is to write with the appropriate audience clearly in mind.

In this phase, the technical writer needs linguistic tools, such as a terminology consistency checker, a grammar checker and a spelling checker which reports syntactic and stylistic errors. Well established and useful tools are available on the market, but they only support revisions at or below the level of the sentence and not across sentence boundaries. Therefore, we claim that a discourse information tool is necessary. So far, while much effort has been put into resolving the grammatical and semantic issues in document production, little is as yet known about the role of the explicit signalling of discourse factors in determining the ultimate effectiveness of the produced document.

J.3.4 Summary

In this section, we have briefly outlined the KBAS-TW scenario, i.e. a sketch of an architecture for a workbench for technical writers, which will support the technical writer in various ways during the process of document production, starting with initial planning, data and information collection, their arrangement in terms of a document structure, the actual document writing, and ending with a ready-to-print document. The project described is still in progress and the final solutions to the different problem areas are not yet fixed in all details.

Nevertheless, our opinion is that this project will have a great impact on the development of efficient and cost-effective solutions and computational means for the process of commercial document production for technical writers, especially because to our knowledge this field has received very little attention in the direction of knowledge-based systems (at least in Europe).

We have focussed on the definition and the realisation of a knowledge base, which provides the various knowledge types for KBAS-TW. Its integration and use in the document production process and its communication interfaces in the overall scenario were briefly discussed. To some extent, we have also demonstrated that the sharing of knowledge sources across projects should be feasible. Based on the different design stages, we are also able to define appropriate evaluation techniques for such systems, which in addition will guide the revision of certain design decisions. We look at this aspect in the following section.

J.4 Towards evaluation

J.4.1 Evaluation methodology

We shall conclude our discussion by showing how techniques for evaluation can be derived from the design stages of one specific system (product) and how they can be combined into an integrated evaluation methodology for knowledge management systems. The goal of the integration is to create a top-down evaluation process that reduces user and data requirements to a standard evaluation structure (ideally interpretable by computers). Such a process shall be called an *evaluation methodology* and contains:

- A sequence of evaluation steps, each one solving one well-defined evaluation problem;
- The techniques to solve the evaluation problem at each step, using the semantics appropriate to that step;
- The techniques to combine all the steps into an integrated evaluation methodology, enabling the output of one step to be used as the input for the next step (as a kind of constraint);
- A documentation method to keep track of all the evaluation results (decisions) and models made during the evaluation.

The evaluation process should be supported by appropriate software tools. These will assist evaluators in two ways. First, they will act as a repository for the documentation generated during the evaluation, thereby enabling evaluators to draw up models, to store them and to change them. Secondly, such tools can help to develop accurate models and to convert a model at one level to that used at the next level of the evaluation process. This is essential for diagnostic and progress evaluation.

Evaluation methodologies are constructed by:

- Identifying the evaluation steps;
- Selecting the techniques and documentation methods appropriate to each stage;
- Selecting the conversion techniques between stages to ensure their continuity.

The evaluation stages must follow the natural design process of a product to be evaluated, which begins by collecting and analysing the data regarding the application that aims at employing the product (purpose of use). The evaluation specification is then prepared and the technical evaluation follows. The evaluator must make the following choices when selecting a methodology:

- The method for collecting information for the evaluation;
- The methods for modelling the evaluation;
- The methods used to convert the model to logical designs (generalisability).

Each choice means selecting evaluation techniques and a method for documenting the output from each evaluation stage.

Documentation serves two purposes in an evaluation methodology:

- It is the medium that presents the decisions made at each stage;
- It provides (administrative) controls for the management of the evaluation process.

Documentation is a record of all the decisions made during evaluation and it contains the final model produced at each stage, which can be used to monitor the project's progress (useful for progress evaluation).

The goal of the data collection is to produce a formal document of user requirements. This document must include the data needed by users as well as the use made of the data. Evaluators should interview the users and then obtain even more detailed information from other sources, such as:

- Documents in the enterprise that wants to use the product;
- Transactions that will be manageable by the product in the enterprise;
- Functional descriptions of business functions;
- Business rules;
- Scenario analyses.

The latter source comprises a detailed examination of the envisaged activities. This analysis differs from the functional descriptions in that it defines, principally, the inputs and outputs of the application's functions. Scenario analysis is thus a deeper examination of the relationships among objects in the function, searching mainly for the activities in the application and building from them.

J.4.2 KMS evaluation

Up to now, we may summarise, a KMS can be evaluated (in a broad sense) according to the degree to which it satisfies the following conditions:

- Ease of use;
- Domain independence;
- Efficient and effective indexing support (machine aided or automatic);
- Response efficiency;
- Presentation of results;
- User and database (knowledge base) services.

These conditions have to be based on specific requirements according to:

User profiles — to permit an adequacy evaluation;

Functionality — to support a diagnosis evaluation;

Maintainability and portability — to permit a combination of diagnosis and progress evaluation;

Hardware and software platforms — to permit a combination of adequacy and progress evaluation;

Communication facilities — to permit a combination of all three evaluation types.

Our investigations in the domain of documentation, however, have shown that user interviews can only help to a limited extent for improving knowledge management systems and information retrieval procedures. This is because users can only evaluate and require what they know and what they can imagine; new application technologies have to be initiated (introduced) first. This is very well demonstrated within the field of *virtual reality* (VR), which, currently, has not received very much attention, although it provides one essential basis of future software and hardware systems in any application area in industry and commerce, education, training, health care, entertainment and the arts (not to mention military applications, cf. the role of VR in the Gulf war).

In this section, we have outlined how the design and specification stages of a workbench for documentation can trigger the evolution of an evaluation methodology. Now, the next step will be to formalise this process, i.e. to define the actual evaluation procedures. This will be the subject of the following phases of EAGLES work.

Appendix K

SdT: A Case Study

K.1 Preliminaries

K.1.1 The genesis of the study

This study has its origins in the work of the EAGLES Evaluation Working Group, a group created in 1992 in the framework of the LRE programme. As part of its work on investigating evaluation methodologies for language industry products, the group carried out a series of case studies of potential users of such products. (EAGLES Evaluation Group Interim Report, July 1994). D.G. XIII suggested that a part of the follow-up work, under the EAGLET extension to the EAGLES programme, should be a more extensive study of the Commission's Translation Service (SdT). That the study was in fact undertaken is due in large part to Roger Havenith, who as a member of SdT working with D.G. XIII had a special interest in the topic, helped to get it started, and provided support and encouragement throughout. This document reports on the study.

K.1.2 Methodology of the study

The study was carried out through a series of loosely structured interviews of a cross section of the personnel of the SdT, including all levels of the hierarchy. The interviews took place between October 1994 and the end of March 1995. A complete list of those who contributed to the study is contained in the appendix.

Before going any further, I would like to record my gratitude to all those who gave so generously of their time: their openness, enthusiasm and willingness to co-operate left me greatly impressed. The organisation of interviews was co-ordinated by Roger Bennett in Brussels and by Carlo Mergen in Luxembourg. They also, with Dimitri Theologitis, provided extensive feedback on the first versions of this report. I owe them a considerable debt. Lastly, I should like to thank those others who read the earlier drafts of this report and provided comment, especially my colleagues in the EAGLES Evaluation Group and in ISSCO, and Roger Havenith, Santiago Del Pino and Jackie Reizer of the SdT. I am of course solely responsible for any mistakes or misrepresentations.

It was agreed with SdT at the beginning of the study not to use a tightly structured questionnaire, on the grounds that, since what was aimed at was a descriptive study of SdT in general, a too focused approach risked biasing the study by pre-judging the topics of interest. Thus, the main aim of the interviews was to listen to what those being interviewed wanted to bring up, rather than to obtain precise answers to pre-determined questions. The interviews were recorded, resulting in about fifty hours of material. However, in order to be able to produce at least a semblance of order from such a mass of material, a set of very general non-directive questions underlay the interview structure.

Each participant was asked to describe his or her own job, and the work-flow, identifying any bottle-necks or major difficulties they experienced. They were asked what use they made of aids to their work, including conventional resource material and resource services or persons, as well as computerized aids. They were also asked if there were any tools which they knew to be available and which they did not use. Before the interviews, a list of tools which were of potential interest had been prepared, and the interviewer tried to introduce these into the conversation when it seemed appropriate. Finally, participants were asked to day-dream: to suggest one or more aids that they would like to be given in an ideal world where technological feasibility, economic constraints and practical or political considerations held no sway.

In practice, it often proved unnecessary to ask all of these questions explicitly: either the conversation tended of its own accord to cover the ground that had been foreseen or a very slight nudge could make it do so.

K.1.3 Status of the study

As may be inferred from the above K.1.2, the present report makes absolutely no claim to objective validity. Less than fifty people were interviewed in all, and no attempt was made to obtain more than an intuitively representative sample. The techniques employed are not such as to lend themselves to statistical analysis, even if the sample had been larger or more representative. The aim here is modest: to present one outsider's picture of an organisation which is unique in its size and its complexity. The usefulness of the report will, it is hoped, reside in its conveying to others just how different the SdT is from other translation services, and perhaps in its providing fertile ground for ideas on how the production of multilingual documents can be facilitated.

It should also be noted that the report represents a snapshot of the state of affairs at one particular point in time, and, like all snapshots of moving objects is unlikely to be entirely accurate. One of the aspects of the SdT that was most apparent throughout the study but especially during the period in which this report was being written was the speed with which the SdT evolves; every new draft brought forth comments that although some point or other was certainly so at the time of the interviews it had by now changed. Where possible, changes that intervened during the report writing period have been incorporated, but it is still true that what is represented is what the author believed to be true on October 20th, 1995. By the time this report is read, there will certainly have been further developments.

K.1.4 Structure of the report

The report proper falls into five main sections.

- The first of these is intended as background (K.2). It seeks to recapitulate the task undertaken by the SdT (K.2.1), its organisational structure (K.2.2) the current state of computerization (K.2.3) and the tools (K.2.4) available.
- The second tries to describe the work of the Service (K.3). It is articulated around the document production chain, and the actors who intervene at various points.
- The third describes in more detail the work of those who provide support to the main activity of translation (K.4).
- The fourth collects together the dream tools (K.5). It should perhaps already be said here that not all of the dream tools enjoy the same degree of fantasy. Some of the desires expressed could, in fact, already be fulfilled by using tools which exist already, either directly or with a small modification to the mode of usage. Others would require development, but no major advances in the state of the art, yet others would require major breakthroughs. All are included rather

indiscriminately because looking at what suggestions were put forward helps to illuminate the current state of awareness of the SdT population. The reader should also be aware of concluding that because the list of dream tools is extensive, there is wide-spread dissatisfaction with what already exists. In fact the contrary tends to be the case: as people grow familiar with what is already available, they begin to see the possibilities for going even further: acquaintance breeds desire, in this context as in others.

- A final section attempts, by way of conclusion (K.6), to summarize the main points of the preceding sections, and closes with a few personal reflections on the work done.

K.2 Background

K.2.1 The texts

In 1994 SdT was responsible for the production of more than a million pages of translation. An analysis of about 300,000 of these pages shows that 73% will be distributed or used outside the Commission itself, that 41.7% result from a legal obligation to produce translations, 30.6% are political texts and 27.7% are to do with operational needs.

Before the accession of Finland and Sweden at the beginning of 1995, there were nine Community languages: Danish, Dutch, English, French, German, Greek, Italian, Spanish and Portuguese. Finnish and Swedish bring this number to eleven. The first regulation adopted by the Council of the European Economic Community in 1958 establishes the principle of equality of all the official languages of the Community. This principle remains in force.

As a consequence, too, the term "translation" as used in the rest of this study, only makes sense as long as a document is internal to the Commission. Once it leaves the Commission, there is no indication in any document, whether it is an official document or the somewhat less official type of material exemplified by brochures or information leaflets, of the language or languages of the original.

Communication inside the Commission tends to pass through three of the eleven languages, English, French and German. This is reflected in the demands made on the translation services, and thus in the number of translators into those languages (around 160 for German, 140 for each of English and French, as opposed to between 90 and 100 for each of the other languages).

The time available for a translation to be produced varies enormously according to the nature of the document. In some cases, a translation may be required within hours, in other cases it may not be required for some weeks. One of the participants, though, pointed out that there are two kinds of urgency. There are documents which by their nature are urgent, and documents which become urgent by being pushed down the queue by others with higher priority.

K.2.2 The organisational structure

At the end of 1994, the personnel of the SdT numbered almost 1,700. Of these, 1,175 were language specialists and 452 support staff.

The Director General is responsible for the overall strategy of the SdT, for human, budgetary and computing resources, for their organisation and for criteria for their distribution. He is supported by a Senior Management Committee, which is the main decision making body of the SdT. We shall return to the Senior Management Committee at the end of this section.

Until 1989, the SdT, like most translation services, was organized around the particular languages. A re-organisation, intended in part to bring the translation services closer to the requesters, resulted in the creation of departments (Groupes thématiques), each of which deals with the translation requests coming from a specific set of customers. The distribution is as follows:

A : General and Administrative Affairs, Budget and Financial Control.

B : Economic and Financial Affairs, Internal Market and Competition.

C : Agriculture and Fisheries, Regional Policy, Structural Policies.

D : Foreign Affairs, Customs, Development, Expansion and Humanitarian Aid.

E : Technology, Energy, Industry, Environment and Transport.

F : Social Affairs, Consumer Service, Human Resources.

G : EUROSTAT, Information and Innovation, Credit and Investments, ECSC (European Coal and Steel Community) Consultative Committee.

The first five of these departments are based in Brussels, the second two in Luxembourg. A Director in Luxembourg is specifically responsible for the functioning of the services in Luxembourg, and in particular for the management of human and budgetary resources there.

Presence in two sites implies considerable travel across the Ardennes, especially for some of the more senior staff. Since the larger part of the service is concentrated in Brussels, meetings tend to happen there more frequently than in Luxembourg. One of the Luxembourg staff interviewed remarked that travelling to Brussels by train with his colleagues at least once a week had significant value in that it offered time for discussion on questions of mutual interest.

It will be realised that the organisation into departments also allows translators to specialize. But it would be a mistake to think that each group works only for a single requester: rather, requests are received from several different organisational sources, which may well have conflicting priorities. As an extreme case, the head of one department said that he dealt with 150 different requesting groups.

Within each department there are nine units, one for each of the languages. A Unit typically consists of between fifteen and twenty translators plus support staff. As we have already noted, the units dealing with English, French and German are larger than the other units.

Finnish and Swedish are beginning to be dealt with inside the Commission, though much work is still done outside. By the end of 1995, about 50 linguistic staff will have been recruited for each language. This will eventually become a hundred translators and thirty secretaries for each of the languages. The new staff will initially be organised into independent sections according to language. They will not be integrated into the departments until somewhere around the middle of 1996, when at least seventy linguistic staff will have been recruited for each language.

A certain proportion of documents (a little less than 10% of the total) are multilingual and multi-thematic. Though these documents are translated within the normal translation units a separate Unit dealing with planning and resources (SdT/01) coordinates their production.

SdT has its own computer services unit (SdT/02), numbering some forty staff. The computer services are responsible for the functioning of the hardware and software infrastructure of the SdT, for the coherence of the different parts of that infrastructure, for training in the best use of the services offered and for assistance to users.

There is also a small unit (SdT/03) responsible for modernisation of work methods. The main aim of this unit is work out and present the options available for how the work of the SdT might evolve.

A number of tasks are concentrated in central units, grouped together into the Directorate for General and Language Matters (AGL). AGL personnel number around one hundred and eighty, of whom around ninety are linguistic specialists.

AGL 1 deals with coordination of problems specific to each particular language, and has a language coordinator for each of the languages. AGL 2 deals with training and recruitment. AGL 3 deals with terminology and language support services, including the promotion of new translators' aids tools, creates and manages linguistic and terminological resources to be integrated into translators' aids tools and collaborates with the other Institutions on exchange of terminology. AGL 4 deals with the specification and development of multilingual tools, analysing and defining needs and sometimes building prototype versions, coordinates the participation of the SdT in the linguistic programmes managed by D.G. XIII and seeks to use the results of those programmes for the benefit of the SdT. AGL 5 deals with freelance translation.

The work of all these units is described in more detail in section (K.4), where support services are discussed.

Apart from the central translation services, established in Brussels and in Luxembourg, there are outposts attached to the Commission Offices in Bonn, Copenhagen, Milan, Munich, Athens, Madrid and Lisbon, whose task is to facilitate liaison with freelance translators. They also help the Offices by translating material for local public distribution and maintain and reinforce contact with translators' schools and academia.

The SdT makes heavy use of freelance translators: about 17% of the total volume of translation was dealt with by freelances in 1994, and this has increased in 1995 to about 20%.

There are also teams in the capitals of the new Member States, whose task is to revise the translation of secondary legislation into the new languages. These groups each consist of 15 language specialists, 5 secretaries and a coordinator.

The Senior Management Committee is the main decision making body of the SdT, and as such, tries to keep an overview of all matters pertaining to the Service. Since its work presents the Service in a microcosm, it is perhaps worth looking at it in a little detail before leaving this section.

The Committee is chaired by the Director General. Its members are the Heads of Department, The Heads of Planning, of Computing Services and of Modernisation of Work Methods, as well as the Director of AGL. Meetings are also attended by the Director General's assistant and his administrative assistant, and by his private secretary who prepares the meeting and keeps the minutes. Non-members may be invited to attend the meetings in order to report on or to discuss particular topics. Fairly regular visitors, for example, are the Heads of the Units responsible for freelance translation, for training and for recruitment, as well as those responsible for the horizontal services.

The agenda for the Senior Management Committee is planned over a period of about two months, so it is often possible to know well in advance when a particular issue will be discussed. The Committee meets once a week on Monday morning. The draft agenda for each meeting is prepared and distributed the previous Wednesday. Modifications can be introduced up until Friday morning.

The agenda is organised around three main areas. First there is a part typical of all meetings, for example approval of the minutes. Also in this section the Director General will report on his meetings with other Directors General, give news from the Commissioners and so on. A second section contains items for discussion. The third section contains items such as written communications for circulation, where members may comment but there is not expected to be any thorough discussion.

Matters discussed in the Senior Management Committee are reflected in the weekly information sheet ("feuille d'information") which is distributed throughout the SdT. This is a double sided page of information which contains the latest administrative information. It will for example report on important meetings between officials of the SdT and external parties or on visits from dignitaries, will give summaries of important documents presented to the Service, report on movements of personnel, give important computing news, make an announcement when a call for tender of interest to the Service goes through and so on. The final point on the agenda of each meeting concerns what should be reported in the information sheet.

Minutes of each meeting are distributed the following Monday. The agenda, the minutes and most of the documents which have been discussed are distributed in the documentation dossier (the "liasse") of the Senior Management Committee, which, in principle, reaches every individual in the Service (documents in a very preliminary early draft state are not included). The documentation dossier is distributed the day after the meeting to all participants, with extra copies to the Heads of Department for distribution to their Heads of Unit. At this point, distribution policy will vary: some Heads of Unit pin the dossier to a notice board, others will circulate it. Neither policy guarantees, of course, that the dossier will actually be read. Indeed, both the dossier and the information sheet have something of a reputation for being rarely read.

However, the meeting of the Senior Management Committee and the dossier will form the basis of the meeting which the Heads of Department hold on Tuesday or on Wednesday with their Heads of Unit. We shall take this up in more detail in a later section when we look at the work of the Heads of Department and of the Heads of Unit. Enough has been said here to give an overview of the organisation of the SdT.

K.2.3 Computing equipment

The SdT is in the process of migrating from a UNIX-based mainframe system where users were supplied with dumb terminals to a local area network platform where PCs are linked to UNIX servers. In fact, this migration is almost finished: the remaining three hundred terminals will be replaced by PCs over the course of 1995. PCs are distributed to the SdT staff on request. At the end of 1994, 800 PCs were already in service in Brussels and 400 in Luxembourg. Almost all are equipped with Windows interfaces and connected to the local area network. The network in Brussels and the network in Luxembourg are linked through IDNET.

In view of the accession of the new Member States, a skeleton computer environment was installed in the capitals of each of them in 1993, in order to allow the teams working on the revision of the translation of the secondary legislation to be adequately equipped. These environments were enlarged in 1994.

K.2.4 Computerized tools

K.2.4.1 Document preparation tools

In what follows and throughout this report, tools is taken in its widest sense to mean any tool which may help the work of SdT as a whole. Thus planning and management tools (K.2.4.3) are included as well as tools more directly related to the production of translations (K.2.4.5).

One item, which is not strictly speaking a computerized tool nonetheless plays an important role in facilitating computerization of text handling: this is the EUROLOOK standard for document preparation, which is intended to ensure a uniform appearance of texts and whose use also facilitates convertibility between the different text-processing systems in use. (Note that some of the requesting services use Word as their text-processing system, rather than Word Perfect, which is standard throughout the SdT).

At the end of 1994, the following tools were in use.

Word-processing Word-Perfect 5.2, integrated under Windows is available on the PCs, Q-One on the terminals which are being phased out.

Spelling checkers spelling checkers are available for all Union languages.

Grammar checkers the set of checkers that come with Word-Perfect 5.2 are available, but are only installed on request.

Preparation and manipulation of tables EXCEL

Converters DIALOGIKA on UNIX/PC.

Preprocessing the term preprocessing is used to mean automatic replacement in a source text of terms and expressions by their translation in the target language. On the UNIX servers using the Q-One text-processor, preprocessing was done by a programme "replace" which uses script-files, bilingual files manually created in the Q-One editor. On the PCs, preprocessing is done using TMan, a piece of software created by AGL 3, to manage bilingual or multilingual lists of terminology/phraseology for automatic replacements in Word Perfect files. This limited distribution (c. 50 installations) tool is also used for pre-Eurodicautom (q.v.) terminology processing and computer-aided repetition analysis and multilingual concordancy of Wordperfect texts, primarily by AGL 3. Certain highly repetitive texts, such as the monthly Bulletin of Commission activities, are systematically subjected to preprocessing as soon as the source texts arrive. TMan is also being used by AGL 3 and the Secretariat-General jointly in pilot operations for the integrated multilingual production of structured texts.

Document comparison software DOCUCOMP (on some PCs). Document comparison software is important for the management of documents which may come in successive different versions. The problems associated with version control will be further discussed later.

K.2.4.2 Communications tools

- Commission-wide X.400 e-mail is available
- some free-lance translators communicate through e-mail although most use diskettes.
- e-mail with external partners is available and is being increasingly used
- access to internal and external data bases through IDNET and X25
- POETRY: a Commission specific software package which allows electronic exchange of documents with some requesters. POETRY is in operational use in the Spokesman's Service and in the Secretariat-General's unit dealing with the General Report and the Bulletin. It is in pilot phase in D.G. V in Luxembourg, in D.G. VII, the Consumer Policy service and in the Publications Office. POETRY is also integrated with the Secretariat-General's "Parliamentary Questions" system, and requests for translation have been automatically exchanged since the beginning of 1994.
- EURAMIS a Commission specific integrated interface to machine translation services (SYSTRAN, q.v.), to batch access to terminology (EURODICAUTOM, q.v.) and to legislation titles (CELEX, q.v.). This interface is currently under test in the Translation Service with a view to distribution throughout the Commission. Available on PCs.

  There is also a project of the same name under way, managed by D.G. XIII in cooperation with the SdT, which aims at the intelligent integration of existing resources and the development of additional tools, such as alignment tools, concordance tools and databases of linguistic resources, including translation memories.

K.2.4.3 Management tools

- SUIVI: a Commission specific software.
  SUIVI I Personnel is an aid to personnel management, and contains staff records. SUIVI II Demandes de Traduction records the passage of a translation through the service. It is used to monitor work-flow and is the source of statistical information about the SdT's work.
- APEX: a Commission specific aid to the management of free-lance translators and typists, dealing with both dossiers and contracts.

- DOTBIB: a bibliographic data base recording personal and library holdings.
- ACQUI: acquisition of library holdings.
- GESPER: circulation of periodicals.
- TRADOC: translation documentation (soon to be replaced).
- ASSYST: management of computing equipment and of the computing help desk.
- FORUM: management of computer training.
- Management of multi-thematic documents.
- Management systems of other D.G.'s: SYNCOM, SYSPER, Missions, Official Journal.
- Local management tools on the PCs.

K.2.4.4 Translators' aids

EURODICAUTOM central terminology data base. EURODICAUTOM is very heavily used: to give an idea, there are around 3'500 accesses to it a day.

PREDIC An ad-hoc system for the preparation and uploading to the EURODICAUTOM mainframe of provisional (local) terminology.

TERMI A server-based local terminology data base of limited functionality due for imminent replacement.

SYSTRAN machine translation system. SYSTRAN is maintained and developed by D.G. XIII with assistance from the SdT. Any user inside the Commission and some authorized users outside the Commission may have access to SYSTRAN by e-mail. The user may obtain a raw SYSTRAN translation with automated search for references in the CELEX legislation data base, or a bi-lingual list of terms generated from a text by the SYSTRAN dictionaries. It should be noted that the EURODICAUTOM terminology is now integrated into the SYSTRAN dictionaries. There is also a service used mainly by officials outside the SdT whereby raw SYSTRAN translation is post-edited rapidly by free-lances. It is also possible for a user of the search system for Parliamentary questions to request an automatic SYSTRAN translation of the answer to the question.

Recent policy has been to concentrate on the development of SYSTRAN as a tool for translating the less disseminated languages into German, English and French. Along the same lines, there are Contracts of Association with some national administrations, e.g. the Greek government, for the development of certain language pairs.

DOCUMENTS classification scheme for translation dossiers.

COM/SEC/C documents References for translations. The abbreviations refer to the numbering schemes used for documents. COM documents are Commission documents, SEC documents are proposals for legislation, C documents are Council documents.

Documentary data bases CELEX, ACTU, ECO1, PRC ...: of these, CELEX, a data base of legislation, court judgements and preparatory documents is of primary importance.

External data bases some fifty external data bases are available, including, for example, data bases of specialist and general newspapers and reviews.

It is perhaps worth noticing that of all the tools available, some are used much more heavily than the rest. EURODICAUTOM and CELEX in particular are frequently consulted and much appreciated.

K.2.4.5 Multilingual generation tools

- AVIMA: this tool is used by the Publications Office, but has an important effect on the SdT's work by providing for the automatic treatment of a large amount of text. The processing system was created to deal with one specific problem, the translation of calls for tenders. The body of this material has grown with the growth of the Community. Initially there were 1'000 calls a year to be translated into 6 languages: three years ago 20'000 calls, and in 1994 90'000. At the same time, the number of languages has grown first to seven, then to nine and now to eleven. Dealing with this volume of work through conventional translation is simply not feasible.

  In order to find a solution, much of the material has been standardized. First a standard level of information was decided upon, and then a nomenclature specifically created with translation in mind defined. The first nomenclature dealt with public works, and nomenclatures for supplies and services were subsequently added.

  The system allows documents to be keyed in through codes, directly in the Publications Office. Text in each of the languages is generated directly from the codes. Consequently, a team of about 80 people, mainly involved in document preparation but also including a few translators, succeed in generating a million pages of multi-lingual text a year.

  By the end of 1995 a full nomenclature will be available and the whole of the translation process will have been automated. (At the moment, a brief summary is still translated by conventional means).

- TMan: as noted earlier, TMan (K.2.4.1) is in use for pilot multilingual text generation operations involving AGL 3 and the Secretariat-General's Unit responsible for the Bulletin and General Report.

K.2.4.6 Maintenance, development and future plans

All of the computing machinery and tools (K.2.4) listed above of course require maintenance. Many of the tools are also affected by the change from UNIX and Q-One to PCs with windows and WordPerfect; their adaptation is either completed or under way. Other tools evolve with time or the changing needs of the environment. Here, we simply pick out some of the important developments planned for the near future. This section will also begin to take us into some of the problem areas connected with present and future functioning of the SdT.

POETRY Some modifications are needed to the pilot installation currently in use, after which it is planned to install the system throughout most of the Commission services. Building up a large electronic corpus of texts and translations is an essential pre-condition for the use of many new translation tools, of which translation memories are only the most obvious. Widespread use of POETRY will contribute to the creation of such a corpus.

However, encouraging the exchange of documents by electronic means does raise some problems. First, it is important that requests going through POETRY are also recorded in SUIVI (K.2.4.3). This is linked with one of the very common work-flow management problems caused by working with electronic means: because there is no physical object in the form of a pile of paper, people forget to send requests or send them twice, and similarly forget sometimes that a document has been received. The solution here is a change in working habits, but experience with other electronic tools shows that the change can take time. Another problem is the validation of authorisation of a request for translation: signatures are not easily sent by electronic mail. On a more banal level, printing and photo-copying may lead to organisational problems. On the non-electronic system, requesters prepare as many copies of a document as the number of languages they request translations into, plus one extra. With an electronic system, this burden is passed to the translation services, who must print out and copy the documents. This problem is aggravated by the difficulty experienced in the SdT with keeping secretarial staff. The problem would, of course, disappear if all translations were done completely electronically. But several of the translators interviewed pointed out that even if they work on the screen, they like to have a paper copy of the translation at their side to refer to, partly because scrolling backwards and forwards on the screen is tiresome and inefficient, partly in order to have an overview of the text easily available, partly because looking at a screen all the time is very tiring.

MULTIDOC It is intended to integrate a number of existing documentary data bases (ACQUI, DOTBIB, GESPER, TRADOC, PC-BIB).

Document Server This is a project to create a document server. In fact the server will be a virtual entity made up of all the hardware, software and organisational tools needed to manage electronic documents within the SdT. Eventually, the document server will be a repository for all original and reference documents, as well as for all translations.

A very frequently mentioned problem with the current functioning of the SdT should be mentioned in connection with the document server. A document receives an identity when it arrives in the Translation Services and is given a translation number. In some cases, this number disappears when it leaves the services and is replaced by a COM, SEC or C number. Moreover, a single SdT document does not always correspond to a single COM or SEC document: a COM document may very well consist of several SdT documents put together. The contrary may also sometimes be the case. Furthermore, the text and its translations may subsequently be modified, by the requester, by the Legal Services or by the Council. Sometimes, but by no means always, the modifications may come back to the Translation Services. If they do not come back, the translations stored in the SdT archives do not correspond to the definitive version which is eventually published in the Official Journal. The translation may in any case be difficult to trace because of the change in numbering. The COM, SEC, C tool mentioned earlier is a partial response to this problem, but the only way to be absolutely sure that the definitive version is available for future reference would be to have access to the Official Journal versions, and perhaps to be able to search them by key-word rather than by document numbers which are susceptible to change. Full-text search would also be useful. Unfortunately, making the text of the Official Journal available in electronic form for archiving and retrieval involves a certain number of legal problems, since the Publications Office has separate contracts with a number of different publishers for publication of the Official Journal. CELEX (K.2.4.4) makes the official versions of some documents available, but the problem was nonetheless mentioned often enough to suggest that this is still felt to be a pressing problem.

It should be noted too that the practice of modifying translations after they have left the Translation Services can be a source of considerable frustration for the translators, who sometimes see their work "corrected" in such a way as even to introduce grammatical errors. And since the published version is the official version, this can also lead to having to quote the offending passage!

SEI-BUD This is an integrated document preparation system for the draft budget proposal. Three main actors are involved: D.G. XIX (the authors), the Publications Office (printing) and the SdT.

The document consists of some 1,700 pages, and has up until now been prepared literally by using cut and paste on paper to introduce the modifications which take place each year. To increase the burden, even in the literal sense, the operation was carried out on A3 paper. The whole text was then re-typed in the Publications Office.

A prototype of the system allows the whole to be stored in a central repository as a single document, in the form of "editorial objects", coded in SGML. Each object contains a segment of text. Other information such as what part of the whole this object is, what language it is in, what version it is, who is working on it and so on is associated with the object. A set of filters allows the various actors to use their own tools: WinWord, Word-Perfect, Excel, Interleaf. This avoids re-typing, and also allows translation of parts which are unlikely to change to proceed without waiting for the whole document to be finished.

The system was successfully put into operation for the first time in 1995. Reactions from the three main actors are positive.

D.G. XIX had to change their working habits from working on paper to working on an electronic document, and had to hire auxiliary typing help to get the data entry done. Nonetheless, the advantage of being able to dispose immediately of a clean electronic copy for internal distribution outweighed the minimal rise in cost.

For the SdT, the improvement was very noticeable. It proved possible to meet the deadlines easily, and since the document entered mainline production, it caused no major perturbation to the normal functioning of the service.

The Publications Office suffered most problems, primarily due to the printer not being able to adapt to processing SGML files rather than introducing all data from scratch. Consequently, the time savings which had been foreseen did not materialize, although the deadlines were kept. Experience should help to iron out most of the difficulty.

The system will now be extended to include the other two Institutions, Council and Parliament. It will also serve as a model for a new project, SEI-Leg (for legislation) to provide for a central repository of Commission documents being worked on.

All in all, the experience proved the usefulness of stocking documents in SGML, thereby rendering them immune to different word-processing systems used by the different actors and to changes over time coming from successive versions.

APEX APEX is the management system which is used to co-ordinate free-lance translation. (We have already noticed that at the end of 1994 about 17% of the total volume of translation was carried out by free-lances, and that this percentage has grown to about 20% in 1995. In the longer term, the proportion may be as much as 30%).

The procedure for selecting free-lance translators has recently changed. Previously, most translation units had a group of free-lances with whom they worked regularly. The free-lances were almost “distant colleagues”, who became used to the texts dealt with by a particular service and to Commission procedures and in-house jargon as well as to the specific terminology of the work of the unit.

However, the volume of work passed to free-lances became so large that a new procedure, aiming at transparency in the selection procedure, had to be put in place. Jointly with the Parliament translation services, a call for expressions of interest was published. The three thousand or so individuals or agencies who responded were asked to translate a brief text for each of their languages, and those whose work was satisfactory are now recorded on a central register. This implies that any translation unit in the SdT can, subject to price constraints, call on the services of any free-lance, as can also the services of Parliament. We shall see later that the change in procedures has given rise to some disquiet amongst the Heads of Department and those responsible for central or group planning. Here we simply note that the change in procedure implies re-design of APEX.

SUIVI Follow-up, the re-design SUIVI, the work-flow management system used to manage the passage of documents through the translation services, is well under way. One feature of the re-design is to permit information to be down-loaded into local versions to be used by the translation units in order to facilitate their own day-to-day management.

Page Counter This project will investigate the use of a page counter. This question is not so straightforward as it may seem to the non-translator.

First, all pages of translation are not equal. Some texts are a great deal easier to translate than others. For example, an experienced translator, dictating, and working with a type of text with which he is very familiar, has been known to produce up to fifty pages of translation a day. The same translator, working on a different kind of text may be lucky if he produces one page, especially if he runs into problems of terminology which require considerable research or if the author of the original does not write the most lucid prose.

Then there is the question of what exactly to count: many documents come in successive versions, and only the modifications have to be translated from one version to another. If a three hundred page document contains five minor modifications, it would be mis-leading, to say the least, to say that three hundred pages were translated. On the other hand, it might also be misleading to count only the volume of the modifications: translating one sentence may involve reading the whole section in which it is embedded or even more to get a sufficient context.

Translator's Workbench Systems A number of commercial products are currently being evaluated within the Translation Services.

Once again, though, the particular work context of the SdT creates special needs. The products currently on the marketE.2 are primarily oriented towards a single-user environment, where one translator gradually builds up his own archive of texts. Another factor in a large translation service is that staff sometimes suffer from frequent interruptions in their work. Not all tools tolerate interruptions well: sometimes a user has to finish a job completely or abandon it and re-start from scratch, instead of being able to leave it for a while in mid-task.

Within a very large translation service, creating a translation memory is far from being straightforward. It would be naive to think that any translation done could simply and automatically be used to feed the translation memory. It may be that the new translation is not consistent with previous translations of the same or very similar text elements, and that in fact the previous translations are preferable. It may be that two people in two different departments are simultaneously translating very similar texts; the probability that they will produce the same translation is very low, and adding both to the archive may unnecessarily increase the amount of redundant material retrieved when the memory is consulted. It may be that the new translation is subsequently to be revised, and that it would be inappropriate to archive it before revision. If the reviser is not working on an electronic copy, there may be difficulties in any case in capturing his revisions. Thus, there is a series of questions to do with validation of the translation which have consequences for how the system can be used.

One suggestion was that it would be helpful to make a strong distinction between an archiving system, a working system, and the individual's own previous copies.

There are also, of course, all the storage and retrieval problems associated with a translation service that produces more than a million pages of translation a year. It is worth mentioning that one of the people interviewed pointed out that there is already a memory problem. Documents cannot be kept in electronic archives allowing easy access indefinitely, because there is not enough space. It sometimes happens that a document which has been removed from the live archives to tape archives is needed. It is possible to get it with the help of the computer staff, but that can sometimes take more time than is available.

Local Terminology Tools Again, a number of tools for use on the PCs to allow a user to create and use his own local terminology base are under evaluation, and individual users or groups of users have already created local terminology bases on their own initiative. Intuitively, such local terminology bases are a valuable potential source of material for the central terminology resources. But questions of validation very similar to those raised above in the context of translation memory systems arise here too.

With terminology other additional factors affect the issue. Several of the people interviewed mentioned that translators were sometimes reluctant to share terminology. This is not only because of a proprietory attitude to the fruit of their own labours. A translator may know that he has been forced to produce a translation in a hurry, and be somewhat uncomfortable about some of the solutions he has adopted, at least feeling that he would have liked more time to mull the problems over or to do research. In these circumstances, he will obviously be unhappy with any system which automatically seizes his solutions to feed a common resource.

On the opposite side, because any translator is aware of having been sometimes forced by urgency to take short-cuts, he can lack confidence in other people's solutions: they too may, after all, not have had the time to search for the really good solution. He will want at the very least to know the source of the solution.

At the time of writing, a strategy to deal with the practical issues of combining local data with general availability is being discussed. The proposal is that local data should have a structure which is a subset of the EURODICAUTOM structure, so that immediate uploading capability is available, and that individual translators should be encouraged to upload their local files into a central area. Subsequently the local data will be taken from the local area and validated by the terminologists before being included into the generally available terminology. The open question is still, of course, how to encourage translators to upload their material. One way, it is felt, is ensure that updating of the generally available terminology is done speedily, so that the individual translator can see the results of his collaboration made concrete in terms of improved general resources within the space of a couple of months.

New organisational structures A very recent innovation is the creation of new organisational structures within which work on new tools can be concentrated. The aim is to develop and continuously test new tools by using them to translate suitable types of documents, in particular repetitive and/or urgent texts.

Two different but parallel structures are foreseen, a "Translation Workshop" ("Atelier de Traduction") in Brussels, and a "Modernisation Network" ("Réseau de Modernisation") in Luxembourg.

Translation workshop The strategy here is to bring together translators and linguists from AGL (both assigned on a voluntary basis for one year) with some secretarial support to constitute a new task force of some twenty staff. One of the main ideas here is that the volunteers will become specialists in the use of new tools and will be able to pass their expertise to their own unit when they return to it.

Modernisation Network The strategy here is that participants in the network stay in their own units, with a loose structure holding them together. The network is coordinated by a member of the AGL staff. The idea is to check how integration of new tools, mostly pre-processing and translators' workbenches can work within the existing structures of the Service thus avoiding the need to create new structures.

Inter-Institutional Collaboration Inter-Institutional Collaboration touches on three main issues.

The first of these is recruitment and training, where the possibility of organising common entry examinations is being investigated, and where training courses are being jointly organized to ensure that each course can reach the critical mass required.

The second of these is concerned with complementarity in management, the idea being that the translation service of one Institution can help out that of another in time of need, or, for example, during the summer when staff are less available and when work loads are very variable across the Institutions.

The third concerns terminology, documentation and new computer aids. There is considerable activity in these areas.

Any translation service faces a problem of obtaining, validating and stocking terminology. A very large translation service dealing with documents in very many different areas and needing to ensure consistency both over large groups of people and over large numbers of documents faces the problem to an even greater degree. We shall return to some of these problems later.

Here, though, we should mention projects to alleviate some of the problems through collaboration with the other European Institutions. The Council of Ministers' terminology base, TIS, can already be consulted, and an update procedure is being implemented.

More ambitiously, a decision to create a single terminological data base for the European Union has been taken; implementing the decision will involve tackling and resolving a number of quite complex organisational issues.

The main issue in documentation is the creation of a single numbering scheme for document identification. (We have already mentioned the complications caused by COMM, SECC and C numbers and their failure to correspond). This too is quite a complex issue, and will have to be resolved at the level of the Secretary General.

There are also on-going inter-institutional discussions on document archiving and translator's workbenches, which are intended to lead to creating common technical specifications and to joint calls for tenders for translators' work bench tools.

Information access and exchange There are plans to create a bulletin board service accessible to all SdT staff, which will allow them to post notices and exchange information.

Pilot tests of access to INTERNET have also been carried out and are continuing, although on a fairly limited scale.

A CD-Rom server will be put into prototype service in 1995.

K.2.4.7 Distribution of tools

Most tools are distributed freely to those who request them. However, some tools are perhaps best suited to be used by specialists, and there is some feeling that their distribution should be more limited. Examples include some machine-aided translation tools which are not yet totally satisfactory, being perhaps slow and working through an interface which is not very intuitive.

Some tools by their nature require varying levels of access for different people. SUIVI (K.2.4.3) here is an obvious example. Another example is certain data management tools, such as terminology bases, translation archives and so on, where some users will have read-only access, some add but not modify access and so on.

It should also be noted that where a variety of tools for doing roughly the same job are commercially available, for reasons of support and maintenance, a user cannot necessarily have the particular tool he would like, but only the tool which has been centrally approved. As might be expected, people do not always agree with the central decision, and may sometimes buy and install the tool of their choice.

K.3 The document production chain and its participants

K.3.1 Documents, their origin and their creation

Requests for translation originate in the Commission departments: in the Directorates General, the Cabinets, the Secretariat-General and elsewhere. The requesting departments vary enormously in the amount of translation they ask for. D.G. V is the largest single consumer of translations and over the four months of July, August, September and October of 1994 asked for an average of over 10,000 pages of translation per month. At the other extreme, D.G. XVIII asked for a total of 2,811 pages in the whole of the twelve month period from 1st November 1993 to 31st October 1994.

Demand can also vary quite widely over time: D.G. XII, over July to October of 1994 asked for an average of just over 4,000 pages of text. This average figure if considered alone is misleading: in fact, in September over 12,000 pages were asked for, whilst in the other three months, the volume only once passed the 2,000 page mark.

The amount of translation passing through the Translation Service does not reflect the total amount of translation done inside the Commission. It is quite common for an author to organise translation within his own department, for reasons of speed or of convenience. This has obvious consequences for the completeness of document and translation archives and, regrettably, on occasion, for the quality of the translation.

Although requests for translation come from the Commission departments, the documents to be translated may originate elsewhere, most frequently in the Member States. It is perhaps worth remarking that very little can be done to influence the presentation or the medium of texts coming from outside.

When the authors of a text are Commission staff, (and, for that matter, in the case of some outside documents), there is no guarantee that they will be writing in their mother tongue. Also, many documents are not produced by one single author. Documents may be produced by several authors or may contain excerpts from other documents. A direct consequence of all this is that the quality of the original text may not always be equivalent to that produced by a fluent, educated native speaker: there will be ill-turned phrases, clumsy formulations, unwitting ambiguity and sometimes straightforward grammatical and spelling errors. This is one of the characteristics of the work of the SdT which distinguishes it very strongly from most other translation services: translators the world over complain about the quality of the source text they have to deal with, but few are called upon to deal with text written by non-native speakers to the extent that the SdT is.

Some of the people interviewed suggested that the Translation Service might usefully intervene already at the stage when a document is being prepared, serving as language specialists rather than translators. This ties in with a proposal which is sometimes made whereby the Translation Service is decentralized completely and becomes language groups working within the requesting services rather than an organisation separate from the requesters. Attitudes to this suggestion are somewhat mixed. Most of the people interviewed are very aware of a communication gap between the requesting departments and the Translation Service, and would welcome moves to lessen the divide. But the translators in particular appreciate being able to work with colleagues of the same target language, with whom one can discuss problems. Dispersing the translation activity would inevitably mean creating smaller groups, with very few translators into each language. There is also the purely practical problem of being able to cover all the source languages. A translator translates from, typically, two, three or maybe four languages into his native language. In order to provide translation from ten source languages, then, on average three or four translators are required: there may simply not be the volume of work in a particular department to justify the employment of that number. Inconsistency of work load (see K.3) (see earlier) and its mirror image - the difficulty of guaranteeing continuity of service in the event of leave, illness etc., within a very small translation unit - are further problem areas.

A proposal somewhat along the same lines but not posing practical problems to the same degree is already actively pursued by a few departments. This involves the creation of "task forces", groups of translators who co-operate directly with a requesting department in a perspective whereby the activity is perceived more as the production of a multilingual document than as the production of a document in one language and its subsequent translation into others. This mode of organisation is not as yet very widespread, but there are plans to increase its use.

An almost diametrically opposed proposal made by some translators (often coming from the southern countries!) is to decentralize work completely and organize translation through teleworking. Whilst this would have some obvious advantages in terms of improved quality of life for those who prefer it and perhaps also in terms of keeping the translator in touch with his own language, it raises difficult problems of organization, of protection of social benefits and of responsibility. The questions essentially turn around who would be the employer, and who would be responsible for quality control. It should also be said that even among the advocates of teleworking there were very few who did not want some contact with colleagues: they imagined decentralized translation offices in the Member States much more than true teleworking from the individual home, except for those whose thinking in this area revolved around the question of family responsibilities.

Yet another proposal is the creation of an editing pool within the SdT. Problems of linguistic quality as well as of format and presentation would be dealt with here before the documents are dispatched to the translation units. The edited original version would be sent to the requester along with the translation.

In the meanwhile, many of those interviewed suggested that since text-processing systems are used in the requesting services to produce documents (Word and Word-Perfect are the two most common systems), the authors might be encouraged to use spelling checkers and grammar checkers more than they currently do. There are, though, inevitable obstacles to the routine use of these tools. Spelling checkers vary in quality from one language to another, and are, in any case, only as good as the dictionaries associated with them. If very many words have to be added before the system stops coming up with irritatingly large numbers of false positives, most users will prefer to rely on their own ability rather than on the checker. Grammar checkers, in their current state, tend to come up with even more false alarms than do spelling checkers, and the problem cannot be made to go away by adding information to the system. Many of those interviewed used spelling checkers as a matter of routine, but only one person reported using a grammar checker when he was writing in English, which was not his native tongue. Even then, he limited use to documents of two pages or less, simply because the very large number of false alarms made the process too time consuming.

Some requesting departments notify the Translation Service when they begin work on a major document which will eventually have to be translated, but this seems to be comparatively rare. A very common perception from the SdT side is that authors simply do not think about translation: thus they have an understandable, but unhelpful, habit of finishing a document and then suddenly realising that it has to be translated, rather than taking the need for translation into account from the start, notifying the SdT of their document production plans and perhaps sending certain parts of the document, for example the appendices, off for translation before the whole document is complete. One of the people interviewed suggested that some of these problems might be alleviated if newly appointed officials spent a month with the SdT before taking up their normal duties in the requesting services. Although this is probably not a feasible proposal, it does highlight how strong the awareness of problems associated with work-flow is. Indeed, we shall see in the section on dream tools that several of the suggestions made seem to corroborate a suspicion that work-flow is one of the major bottlenecks, not only in the SdT, but in the Commission as a whole.

An exception to all this is the multi-thematic documents which pass through the hands of the central planning unit of the SdT. These documents are always announced, and many of them, the General Report for example, are even predictable.

Some documents, but by no means all, prepared in the requesting departments conform to the EUROLOOK standard. However, EUROLOOK was introduced comparatively recently, and its use is still spreading.

K.3.2 Translation requests and transmission of documents

The majority of documents are still transmitted to the Translation Service on paper. The requester makes as many copies of the document as the number of translations he is requesting plus one, fills in a translation request for each language, and sends it to the group responsible for supplying his translations together with a request slip prepared using a manual typewriter. The request slip will specify, amongst other things, a date by which the requesting department would like the translation, a contact person in the requester's department, and the purpose of the document, for example if it is needed for a particular meeting on a particular day.

Some requesters make heavy use of fax transmission, perhaps, as one of the Heads of Department suspects, in order to emphasize the urgency of the request. This tends to be disliked by the recipients because of the frequency of transmission problems and the poor quality of the faxed documents. It also shifts the burden of making copies from the requesting department to the Translation Service.

One outstanding exception to this pattern has already been noticed. The Secretariat-General sends the monthly Bulletin in electronic form and also makes heavy use of electronic means for the treatment of Parliamentary questions. Another exception is the communications between D.G. V and Department F. Department F is based in Luxembourg, whilst most of its requesters are based in Brussells, and is the only group working primarily with geographically separated requesters. As we have already noticed, D.G. V is also a major consumer of translations. Thus it was perhaps natural that this communication should serve as the pilot test for the POETRY (K.2.4.2) system, which encourages exchange of documents by electronic means. Even so, of the 7,696 documents sent for translation by D.G. V in 1994, only 1,118 were sent by electronic means.

It should be remembered that documents originating outside the Commission arrive almost exclusively on paper.

It is also worth jumping ahead a little at this point to remark that all translations are returned as electronic documents, no matter what form the originals were received in, partly in order to encourage the use of electronic submission. This can cause problems: some of the requesting departments are not very conversant with e-mail and its use, and it has been known for a requester to be unable to print out the urgent document he has been sent.

When a document is submitted for translation, it should be accompanied by any reference material relevant to the translation to be done. If translation is required of modifications of a previous document, the modifications should be marked and the previous document sent.

These two conditions are frequently not fulfilled. Several of the people interviewed emphasized the importance of thinking in terms of a translation dossier, rather than of a single document to be translated.

Some documents are submitted for machine translation by SYSTRAN (K.2.4.4) and are not transmitted to the Translation Service at all. SYSTRAN translation is available on e-mail through a very simple procedure and is increasingly used by officials in the requesting departments for rapid translation of information-only documents. In 1994, SYSTRAN was responsible for 13,550 translation requests for a total of 140,000 pages.

Documents submitted for SYSTRAN translation are necessarily in electronic form. This leads to a limited amount of scanning-in of documents. Scanning is not otherwise used, partly because the paper originals may be not easily legible and may also contain reception stamps or other marks which complicate interpretation by scanning software. It was also remarked during the interviews that desktop publishing systems and the use of proportional fonts have made scanning a more difficult problem than it was, so that even highly legible, well-produced documents may produce unsatisfactory scanning results.

One suggestion offered was that systematically spellchecking documents before they were submitted to SYSTRAN would improve the quality of the output.

The SdT staff are not always totally comfortable about uncontrolled use of SYSTRAN. One person pointed out that it has been known for unedited raw SYSTRAN output to finish up in the Official Journal, to the embarrassment of the SdT who are perceived as being responsible for all translation.

K.3.3 Reception in the translation services: planning

A request for translation will go either to the central planning, in the case of multithematic documents, or to the department associated with the particular requester. In either case, the first operation is to see how the request fits in with planning. A first problem here is that requests come from a variety of sources, each of which has its own priorities, which may well be in conflict with those of other requesters. There is no central policy about how to decide conflicts: it is left to the person responsible for planning to make decisions based on experience and on general knowledge of what the current important issues are.

The decisions taken may well have an impact on work which has already been accepted and is waiting for translation: an urgent document coming in may push less urgent material down the queue. Not surprisingly, this can sometimes lead to strained relations with the requesters. If someone sends a well-prepared request for translation in plenty of time, it is very hard for him to understand that the deadline still may not be met because of other work coming in.

This problem is much worse for certain departments than for others, especially for those dealing with a lot of political documents or with privileged requesters, such as the Secretariat-General, whose requests have to be given very high priority.

Exactly how the planning is organised varies from department to department. In some cases a document will first be examined to see what reference materials are required and whether they are available, whether parts have been translated before, and whether modifications have been marked. In other cases, the assistants responsible for planning will do this work themselves, and in yet others, some of this work will get passed down the chain to the translation units or even, occasionally, to the individual translator. Sometimes this may happen because those earlier in the chain do not regard it as part of their responsibilities or as lying within their competence. In other cases it may be unintentional: it is not always possible, for example, to determine what all the pertinent reference material may be without subjecting the text to the sort of close examination it will only receive once it is being translated. A very negative side effect of such work being passed down the chain is that the same requester may be contacted by several people (the translators from the different units) to ask the same question or request the same reference document. Clearly, this does nothing to improve either relations with the requesters or efficiency.

One extreme case of the difficulty of finding reference material will help to illustrate its importance. One department deals fairly frequently with "Common Position" documents, documents of only two or three pages which report on the position of all the different Institutions. Consequently, a single document may contain fragments of texts from all the different Institutions, and the exact wording must be reproduced. One case was cited where a document of this type had almost sixty reference documents associated with it. It can take a considerable time just to find the references, and perhaps involve requesting help from the Parliament translators who may already have done the work of collecting the references together.

It was suggested during the interviews that it might also be possible to do some work on terminology when a document first arrived in the Department, rather than leaving it to the individual translation units.

Where documents come in successive versions, some groups use document comparison software to determine what modifications have been made. Others rely on visual scanning. Exasperation with requesters who fail to mark modifications was frequently expressed. It is possible too for mistakes to intervene at this stage. For example, imagine that a document has previously been translated and sent back to the requester. The requester then changes a figure somewhere in the document. Subsequently, a whole paragraph of text is modified, and the requester sends this back to the translation service marking the paragraph, but not the figure. The translation service then works on the first version of the translation, which is still available electronically and sends back a new translation. The modified figure has now been changed back, or rather has never effectively been changed. There is a high probability that the requester will not notice.

Version control and avoiding doing the same work more than once are a major preoccupation for almost everybody who was interviewed. The problem is of course compounded by the sheer size of the organisation. Many people emphasized the importance of good electronic archiving facilities, and the possibility of being able to carry out a key-word search over large archives of documents was one of the most frequently mentioned dream-tools. A Word-Perfect facility of this type, QUICKFINDER, does exist and is used by some people, but its usefulness is limited by inability to handle the complete (and very large) archives in the SdT's networking environment. The issue of access to the Official Journal versions of texts, previously mentioned, is also obviously pertinent here. The planned document server (vide Document server, section 2.4.6) is designed to mitigate this problem.

As might be expected, those responsible for planning make heavy use of SUIVI.

K.3.4 The department heads

The Heads of Department in charge of each department are very closely associated with planning. Their major responsibility is to manage the work requested of them, maintaining contact with the requesting departments and negotiating deadlines with the requesters. Several of them regard communication both with the requesters and within the department as being a major issue. In line with this, the telephone was frequently said to be the tool most used by the Heads of Department.

One topic which came up in discussion with the Department Heads and with the Unit Heads is the idea that not all translations are equal. The purpose to which a translation is to be put might be held to have a strong influence over the attention that should be paid to producing it. Thus, in certain cases it might be appropriate to respond to a request for translation by offering to make a quick oral translation or to give a summary, or by suggesting to the requester that he might use SYSTRAN, either as raw translation or by using the rapid post-editing service done through freelances. This idea sometimes arouses controversy, especially when it is expressed in terms of asking a translator to do a poorer job than he is capable of, which is seen as demotivating. On the other hand, it was said that the SdT needed to become more aware of the political process inside the Commission services, and that being flexible in the way that demands were responded to would help to attenuate a dissatisfaction sometimes felt with respect to the Translation Service's ability to meet the political needs of the requesters.

Another side to managing the burden of work requested is to encourage standardisation of documents and of their presentation both in the requesting services and in the department itself.

The Department Heads meet the Director-General, who is responsible for the functioning of the whole service, once a week in Brussels for the meeting of the Senior Management Committee, and transmit information from this meeting to their Unit Heads. Most consciously make a very strong effort to create a feeling amongst the Unit Heads that the whole department is working together as a single entity. One Department Head remarked that the communication issue was affected by the fact that each translation unit tended to reflect the culture it sprang from. On the other hand, several of them emphasized that once work had been passed to the translation units they tried to limit intervention to areas where they thought it indispensible, such as encouraging modernisation. The use of e-mail for transmitting information within the group was mentioned, although few heads reported using the possibilities offered intensively or systematically.

Many of those interviewed, both Department heads and others, talked about a general problem of ensuring information flow. We have already mentioned the distribution of the dossier of the Senior Management Committee and the Information Sheet which contains a summary of the main administrative news. But busy people whose lives are dominated by floods of paper will not always take the time to read general information, and may even miss or ignore specific information deliberately directed at them. Some alternative measures have already been put in place; for example, anyone who wishes may install a file in the computer network sub-directory "diffusion", and anyone can access the files in that directory. This facility does not seem however to be very much used, and it may be that it shares with ftp nodes a mysterious resistance to use. One current project is aimed at providing a web interface on all the PCs, which would not necessarily give access to information external to the Commission, but would make it easy to find any information within it. This may enjoy more success: it has been a fairly common experience elsewhere that people enjoy working the web. Another current project that has already been mentioned is the provision of a good bulletin board service. This too may go some way towards alleviating the information flow problem.

All the Department Heads, directly or indirectly, make heavy use of SUIVI. Some consult it regularly themselves, and make heavy use of the statistics to be obtained from it. These statistics can be organised in different ways by using different search keys, and this was frequently mentioned as a valuable aid. Others leave actual consultation and updating of SUIVI to their planning assistants, but nonetheless rely on the information contained in SUIVI. A few Department Heads use SUIVI not only to check on the workflow through their own group, but also to see what is happening in other groups.

Also in this context, it was suggested that who should have access to what information in SUIVI should be reconsidered, and access possibilities enlarged.

One possibility which has been suggested outside this study, although no one interviewed explicitly brought it up, is to allow more communication between the SUIVI which deals with staff records and the SUIVI which deals with workflow. This suggestion meets with some resistance from the Permanent Delegation of Translators, a staff organisation which deals with relations between language specialists and management. The main motivation for reluctance is related to the page counting issue mentioned earlier: if the number of pages produced by an individual translator goes down, this may be due to any one of a number of reasons, including difficulty of the translation, the language pair involved, unavailability of reference material, irregular work flow, absence for illness or for attendance at training courses, personal problems and laziness. Translation units are quite small: the unit head will normally know the reasons behind an apparent drop in productivity and be able to take appropriate action if necessary. Someone outside the immediate work context does not have access to the information needed for an informed judgement, and someone who understands nothing about the nature of translation may leap to too hasty conclusions.

The other main responsibility of all the Heads of Department is to ensure that translation, where necessary, proceeds in parallel into all the languages requested. The longer a document is, the harder it is to keep work on it progressing at the same rate in all the languages, especially as all units are usually working on more than one document. In the classic bad case, one unit will finish one document first, a second unit will produce a different document first, a third yet another and so on: the requester needs all versions, so for him all these documents are late. Paradoxically, within the translation units it is often the most efficient who find it hard to grasp that the department as a whole goes at the speed of the slowest unit, and is judged accordingly.

Otherwise, as might be expected, all Heads of Department experience the consequences of the kinds of texts their particular department deals with. Some, as we have already noticed, are expected to deal with a constant avalanche of urgent documents which can make planning a nightmare, others have to deal with extremely technical translation in a variety of fields where it is not always easy to find good technical advisors. Once again, an extreme case will illustrate this latter problem. The department dealing with documents on the internal market frequently has to translate proposals for mergers. These documents, each of about 50-60 closely typed pages, have a short turn-around time, about five days, are extremely technical and can deal with anything from motor vehicles to female hygiene. The documents are always highly confidential. When expert help is needed, consulting the companies concerned is obviously not permissible. The group relies on being able to find background documents and on being able to consult experts via the requesting department or the language help desk.

All Department Heads are involved in personnel matters within their own group. These include matters such as promotions and the performance review which is carried out for every member of staff once every two years. Staff are graded by their immediate superior on a number of different criteria, such as the ability to work with other people or the ability to work with computers. The grading is discussed with the member of staff in question and becomes part of his personal record.

And, of course, as in any hierarchy, each level of the hierarchy inherits the problems of the level below it; thus, for example, it was the Department Heads who first mentioned the problems associated with keeping secretarial staff, with dealing with freelance work, with encouraging changes in working methods and so on.

In addition to the responsibilities which are common to all Heads of Department, some have specific additional responsibilities. For example, one is responsible for liaison with the East European Countries about the adaptation of their legislation to Union legislation and for getting translations done into and out of the East European languages by freelance translators. Another has the additional responsibility of coordinating the Luxembourg activities of the SdT. This brings with it a certain amount of political work in the form of liaison with the Luxembourg authorities, and has also recently meant heavy involvement in the setting up of a new Translation Centre in Luxembourg, which the European Council decided in 1993 to create as a provider of translations for a number of European Union Agencies, which are dispersed geographically all over Europe (the Environment Agency, for example, is in Copenhagen, and the Training Foundation in Turin). The SdT will be responsible, initially, for technical coordination and specialist support services for the Centre.

Some of the Heads of Department regretted that they did not have the time to keep themselves up-to-date with new tools and translators' aids: it is, for example, difficult for them to be away from their department for the time necessary to attend courses and seminars. A few had solved this problem by encouraging an assistant to follow the training offered, and relying on the assistant to pass on the knowledge thus acquired.

K.3.5 Freelance translation

From both the central planning responsible for multithematic documents and from the departments, a substantial amount of work is sent to freelance translators. Around 20% of the total volume is currently sent out, and it is planned to increase this figure in the relatively short term to 25%, rising ultimately to 30%. The primary reason for increasing the use of freelance translators has to do with flexibility. The work-load on the Translation Service is constantly increasing, whilst staffing levels remain stable. Workloads too, as we have already seen, vary considerably over time: there are periods of heavy demand interspersed with periods when less is asked for. It would not be practical or economically defensible to maintain staff levels permanently at the size required to deal with the peak periods, accepting in consequence that particular departments or units will have other times when they are under-used. New languages coming in also tend to increase the use of freelances, especially initially when in-house translators for the new languages are still being recruited.

The economics of using freelance translators are somewhat harder to determine. No direct comparison of costs can be made, since the SdT has invested heavily in translation support tools and resources. It was even argued by some that use of freelances can be a false economy: even though the direct labour cost may be small compared to the cost of a staff translator, once revision time and support and administrative costs are taken into account, the superficial economic advantage may disappear.

The decision about what to send for freelance translation and whom to send it to is taken by the language unit, in consultation with the unit responsible for coordinating freelance work. The criteria used in choosing the freelance translator include the price being asked, the quality of that person's or agency's work, their past performance and the compatibility of their computer equipment. Freelance translators are provided with instructions on how texts should be prepared. Most return their work on diskettes, although a few use e-mail.

The criteria for what to send varies from group to group. Some documents are highly confidential and can never be sent for freelance translation. Most groups prefer to handle important documents in-house, unless the document is very specialised and needs a particular expertise which is not available in-house. Groups where the volume of work makes it difficult, if not impossible, even to check the quality of freelance translation, much less revise it, will tend to send documents where they judge that no quality control is needed for freelance work.

This tends to suggest that freelance work is automatically assumed to be inferior to in-house work. This is not universally the case: several people made the point that freelance work can in fact be better than in-house work in technical areas, and those responsible for the coordination of freelance work point to a strong need for technical specialists.

However, freelance translators do suffer from a number of disadvantages compared to in-house translators. They work in isolation and do not have easy access to the resources available to the SdT translator. These resources are considerable, covering not only the tools and resources already mentioned but also documentalists, terminologists, language co-ordinators, help-desks for terminology and computing. They are also not as familiar as an in-house translator is with Commission procedures, jargon and terminology. Finally, they are paid in direct proportion to the volume of translation delivered - an inevitable temptation to work at maximum speed, possibly at the expense of quality. In order in part to combat these problems, an investment in training freelance translators has been proposed, The effort involved is seen as an investment in the future for the SdT as well as an immediate way of improving the average quality of freelance work.

Some, but by no means all, free-lance translators have communications equipment which will allow them to consult EURODICAUTOM, which helps to combat some of the problems of isolation.

Some groups try to help the freelance translators by preparing for them terminology, CELEX references and other material. But the degree to which this can be done is clearly constrained by the resources available. A document sent out for freelance translation is accompanied by an information sheet, which gives, amongst other information, the name of the Unit Head and of a person within the Unit to contact in case of problems. The freelances themselves sometimes send questions, often by fax, which are sometimes dealt with inside the group, sometimes transmitted to a terminologist or documentalist. The terminologists responsible for the terminology help-desk also report direct requests from free-lance translators, and, as we shall see, the documentalists report some problems in this area.

Even amongst those who are strong advocates of using freelance translation, these problems are mentioned. For some groups, reliance on freelance translation was a major worry.

Much of this was exacerbated by the recent change in procedure. As the units become familiar with the freelances chosen through the new procedure, some of the worries may diminish. In fact, it was noticeable that more worry was expressed in the early interviews than in those of six months later.

It is nonetheless true that the new procedure, given the number of free-lance translators involved, could not carry out any thorough check of their abilities.

The procedure was undertaken jointly by the Commission and Parliament. A call for expressions of interest three years ago produced some 3'000 replies. 72 language combinations were involved and about 50 specialist domains. This meant that the 1'800 companies and individuals pre-selected on the basis of their reply completed a total of 6'000 tests, done without supervision. As a result of the tests, contracts were made with 1'100 entities, of which about 75% are individual translators. The unit responsible for coordinating freelance work concludes in total around 3'000 contracts a year, many of them quite small.

Effectively, the freelance translators who have been accepted are currently being field-tested by being used. And, as might have been expected, some who did quite well on the unsupervised tests are now producing shoddy work.

When a freelance translation is returned to the coordinating unit it is registered and then sent on to the language unit responsible for that translation, where it is evaluated. Some units systematically revise all freelance translation, others do not have the resources to do so, but all units will try to check anything new or difficult. It is here that the problem of poor translation may become known. If a language unit is seriously unhappy about the work of a freelance, it may ask the coordinating unit to intervene. The language unit submits five pages of the translation it is unhappy with, completely revised with the corrections clearly marked. The coordinating unit asks an opinion from two or three independent assessors and takes appropriate action based on their advice. The action most commonly taken is to send a warning letter to the offending freelance, and perhaps change the specialisation for future work. Normally, only after a second or third chance has been given will elimination from the panel of freelances be considered. The decision to eliminate is taken by a joint committee of the Commission and Parliament, which meets once a month to discuss these cases. Elimination does not necessarily imply total elimination: a freelance may perhaps remain on the list for other language combinations or other domains. Any elimination must be justifiable in a court of law.

Over time, therefore, those producing poor work disappear from the lists. This may result in gaps for some language combinations and some domains, so that supplementary separate calls may become necessary. Apart from any such supplementary calls, a new call for expressions of interest will be issued in about three years in order to replenish the lists. There is also a longer term policy whereby families or categories of documents are identified, and specialised calls are made specifically for freelances to work on each category. Using freelance translators consistently to work on particular classes of documents would have the result of removing the burden of those documents from the regular translation work of the SdT. One might say that the intention behind this policy is to remove a bottom layer of work from the regular circuits by sending it for freelance translation rather than to use freelance translation only to deal with the peaks in the demand.

Several of those interviewed expressed worries about the economics of the new procedure. They point out that, since the freelance list is shared by Parliament, and Parliament tends to produce numerous discursive documents which are comparatively easy to translate, a freelance will choose whenever possible to work on the Parliament texts in order to maximise his income. This may force the Commission services to use the more expensive translators on the list, possibly even spending more on freelance translation than they did before. At the moment, Commission costs for freelance work are about 10% higher than Parliament costs. (It is perhaps worth mentioning here that translation is a completely free market: there is no professional organisation exercising strong influence over the fees requested.)

Others, however, saw the present period as being one of transition, believing that eventually their group would again build up on-going working relations with freelance translators charging a reasonable price who were familiar with the work of the group and had gradually accumulated experience and expertise with the particular types of text dealt with. This group included the Head of Unit who told me of a highly prized technical translator who had in fact tried to change to working on Parliament texts, only to be eliminated from the Parliament list because his general translation work was said to be of poor quality.

K.3.6 Machine translation

The Commission has had a long standing interest in machine translation as one of the ways in which the burden of translation might be alleviated. SYSTRAN was first introduced in the mid-1970's, and has been continually developed ever since, a number of other commercial systems have been evaluated on Commission premisses, and direct support for work on machine translation through the funding of research and development projects such as EUROTRA has been given.

Gradually the perception of the appropriate role to be played by machine translation has shifted. In the mid-1970's, both inside and outside the Commission, it was primarily seen as a tool to be used by translators. More recently, the language engineering community as a whole has come to see machine translation as a parallel translation activity, which may substitute for high quality human translation in those circumstances where lower quality material may be acceptable - for information gathering, for example, or as the basis for routing a document towards the most appropriate person to deal with it. Inside the Commission, the choice of what languages should be dealt with by machine translation has also changed. In the early days, it was thought appropriate to concentrate on the languages which serve as major communication languages within the Commission. More recent policy is to concentrate on developing translation from the less disseminated languages into the three major communication languages. Most officials are multilingual, and have a good knowledge of at least one of the three major communication languages. Machine translation from the lesser known languages facilitates access to material which otherwise would have to go through the translation services, perhaps, in the end, unnecessarily.

In line with these shifts in perception, we have already noted that officials outside the SdT can have free access to SYSTRAN translation by e-mail, and may request either the raw translation or a translation rapidly pre-edited, but not a fully polished translation.

In practice, then, SYSTRAN is used mainly by officials outside the SdT, although the latter is still, with 25%, the largest single user, and although an increasing number of translators do use SYSTRAN raw translation as a basis for their work.

When requests for SYSTRAN translation come from officials outside the SdT, the texts do not pass through the SdT at all, and are therefore not germane to the present part of the discussion except in so far as they relieve the SdT of part of its burden. In this context, we should note that one Department Head reported that when his group was heavily overloaded, he replied to some requests with a suggestion that the requester should submit the document to SYSTRAN.

K.3.7 In-house translation: the unit heads

When it is decided that a translation will be done inside the Translation Service, it is passed to the unit responsible for translation into the language requested. The Unit Head is responsible for seeing that the translation gets done, and coordinates the work of the unit. He may also do preparatory work for the translators, for example by making sure that files are in the right directory and so on. His colleagues consult him with problems and questions. The amount of actual translation done by the Unit Heads varies from unit to unit. Some will intervene to deal with very urgent documents, of which there may be many, others would like to do more but are forced by constant interruption to concentrate on the occasional short translation which can be finished quite quickly. Most do a lot of revision.

When they are translating or revising, there does not seem to be any great difference in working methods between Heads of Unit and translators, so we shall subsume discussion into the section on translating, concentrating here on the special features of the Unit Heads' work.

All of the Unit Heads emphasized their need to know exactly what the state of work in the unit was on a day-to-day basis, and several spoke of a need for local management tools, such as local versions of SUIVI. One had constructed his own management data base using PC tools. A few said that it would be useful to be able to know what was happening in other units, and would like any local workflow management tool to make this possible.

It was noticeable that many of the problems that were mentioned by the Heads of Department and their planning staff were repeated by the Heads of Unit. For example, many Unit Heads emphasized the need for documentation and for electronic archiving, and talked about the problem of version control. Several thought that the secretarial staff could be trained to do more preparation of texts, and mentioned that using the EUROLOOK standard was of great help. Some did not use EUROLOOK because their requesters did not use it. Others returned documents in EUROLOOK whether or not they had received them in it. Some unit heads pointed out that translators often put great emphasis on respecting the formatting of the original document, for example changing fonts or font size so that the original pagination can be kept. This can be important when a document has to be discussed with the author over the telephone. In this context, too, preparing the original documents in EUROLOOK was said to be very helpful. It should be noted though that the idea of separating logical structure and contents by using an SGML type markup language is still quite novel, and still meets with some resistance. It is quite hard to explain to someone who is used to document-processing on a system where he can himself insert all the bold face he requires or change margins and fonts when he wants that there is an advantage in inserting tags instead, especially if he finds that the markup tags tend to make the text on screen less readable or aesthetically displeasing.

When a deadline cannot be met, or when other work coming in changes the priorities which previously existed, the Head of Unit may contact the requesting service to renegotiate a deadline or perhaps to propose a solution other than a complete high quality translation. He may also, as we have already noted, contact the requester to obtain reference material or to check modifications.

Apart from trying to ensure that the translations are done on time and are of appropriate quality, the Unit Heads also carry substantial administrative responsibility. It is their job to plan leave and absences, to propose the marks to be awarded on the staff grading scheme for the members of the unit, to propose plans for further training where appropriate and to make sure that the members of the unit have access to that training, to prepare proposals for promotion twice each year and deal with personnel matters. They also make sure that their unit is satisfactorily provided with terminological support, with documentation and with computer support. They represent their unit, for example in meetings of the department or in language coordination meetings and assist and advise the Head of Department in fulfilling his tasks.

Each Head of Unit has a deputy, who follows the work of the unit closely enough to be able to step in for the Head of Unit when necessary without loss of continuity.

K.3.8 In-house translation: the translators

The translator is the critical actor in the whole multilingual document production chain. Translations to be done are distributed amongst the in-house translators in two main ways.

In some groups, the Head of Unit will check who has dealt with previous versions of the document, who has worked on similar texts or who has spare capacity and will allocate the translations accordingly. In others, the translations are distributed on a self-service system, with the translator himself deciding which document to take. Which system is preferred seems to be as much a matter of cultural preference as anything else.

A recent innovation has been the organisation of workshops within the SdT to reflect on the translation process and how it works. These workshops distinguish three states as part of the translation process:

- the research state
- the translation state
- the revision state

We shall follow this same scheme in organizing the rest of this section.

K.3.8.1 Research

Research essentially involves finding references and reference material and solving translation problems, including the problem of terminology.

A properly prepared translation request is accompanied by all the relevant reference material. However, the translator may discover once he starts work on a document that some reference material is missing. There may also be hidden references: quotations, for example, where the source is not given. His resources for finding missing reference material include the archives of previous documents and their translations, the various documentary and bibliographic data bases, the library and the services of a documentalist and the requesters themselves.

Many translators will search the relevant data bases themselves. Others will ask a documentalist to do it for them. One data base very frequently used is the CELEX data base of legislation, court judgements and preparatory documents. Two problems were mentioned with this data base. The first concerns the data itself. Some of the older entries from before 1989 are encoded in an impoverished character set. A great deal of wordprocessing work is needed to recover the text in its original form. The second concerns the interface available on the older Q-One systems, which is felt to be very unuserfriendly.

It should also be pointed out that, inevitably, CELEX is never up-to-date: there is a delay of up to three months before the contents of the Official Journal find their way into CELEX, and of about eighteen months for court cases. There is a difference too between when the orginal may become available and when translations of it are available. Thus, although a recent decision of the Court of Justice may result in court rulings becoming available more quickly, there may well still be a substantial delay in the translations becoming available.

One way of procuring a translation of the references to previous documents included in a text is to ask for a SYSTRAN translation: an automatic search of CELEX during the machine translation process results in a list of references appended to the SYSTRAN raw translation. Many translators mentioned using SYSTRAN for this purpose alone. This use of Systran will be replaced by the EURAMIS integrated interface currently under test in the SdT, which allows CELEX search alone. Although it is comparatively rare for translators to use SYSTRAN as a starting point for the translation itself, it is also occasionally used to get a rough translation of background documents.

Terminology research also makes up a major part of the translator's work.

A primary source of terminology is EURODICAUTOM, the Commission electronic terminology base. The base can be consulted through a single query or by sending a list of queries. Recently it has become possible to submit a text to SYSTRAN and receive in return a list of terminology extracted from the text, based on the SYSTRAN dictionaries enriched by the EURODICAUTOM data.

However EURODICAUTOM is a long-established data base, and using it reveals some problems. The two most frequently mentioned were the amount of noise in the form of numerous responses to a query, and the problem of knowing what the original language of an entry had been. This latter perhaps merits a little explanation: when a new language is added to the entry, the terminologist will normally be working from one of the languages already there. There is often no indication in the entry which language was the original, although sometimes the reference field may give details which will in effect indicate the original language. But this is far from being the usual case, and so it is possible for a Portuguese equivalent, say, to be constructed on the basis of a French equivalent, which was itself constructed on the basis of an English term. If terminology were such that chaining from one language to another always produced the same result, this would not matter. But unfortunately this is not the case, and the end of the chain may turn out not to be the correct terminological equivalent to the term at the beginning of it. In fact, this is a simplified statement of the problem, which may go further: in some cases, there may not be one single original language, since several entries in different languages may have been taken as the basis for different new entries.

Despite these problems, EURODICAUTOM is very heavily used: almost everybody mentioned it as very high up on the list of tools they used, although I did meet just one person who claimed never to have consulted either EURODICAUTOM or CELEX.

The terminologists and those responsible for EURODICAUTOM point out the same problems, and add the difficulty of maintaining and updating a term bank of this size and age. All the questions of validation which have already been mentioned re-appear in this context, and special problems appear when the possibility of cleaning-up and consolidating the entries already in the base is considered. When EURODICAUTOM gives multiple responses to a query, it may be that some or all of them can appropriately be synthesised into a single entry. This is difficult and demanding work, which takes time. The resources needed are substantial, and are not always available. A recently installed facility which allows consultation of EURODICAUTOM in batch mode with a list of terms may to some extent facilitate the task of maintenance.

The Council terminology bank TIS is also available for consultation and is used. Daily up-dates to it are received.

Many translators create their own terminology data base, either manually on paper or through PC tools. Some units share their terminology across the unit and some groups encourage the creation of requester-oriented terminology. Almost all individual translators consult their colleagues, at least from time to time. This leads to the creation of local terminology resources within individual translation units or departments. We have already mentioned the potential value of such local terminology as material to be included in the central resources, but have also mentioned some of the problems of sharing and of validation.

Some units have established cooperation with specialist organisations in the Member States. They consult them on terminological questions and obtain feedback from them on the translations done.

Some translators make a practice of going at least occasionally to meetings of the Committees which use their documents. In this way they get to know the national experts and can contact them for terminology problems, and can also get feedback on their translations. One suggestion made was that lists of those who had attended the meetings might be systematically sent to the translation units so that they could solicit feedback if they wished. One translator mentioned that seminars organised with the national experts were of great help.

When all else fails (or even, according to the people responsible for running it, before), the translator may appeal for help to the terminology helpdesk, run by AGL 3. The terminologists on the helpdesk point out that there are clever ways to search EURODICAUTOM which help to increase the success rate and to cut down noise. Some of these involve use of the subject codes, which, although necessarily intuitive and therefore open to interpretation, one can become expert in. They suspect that translators become discouraged if they do not find the information they need through simple means, and are also sometimes embarrassed to admit that they do not know any of the tricks. They suggested that it might be helpful to organise short training sessions on clever ways to search EURODICAUTOM.

K.3.8.2 Translation

Ways of working when actually translating vary greatly.

At the moment, most translators work alone. But in some groups, translators are encouraged to work in cross-language teams, at least for some documents. Where this happens, translators find it helpful to be able to look at the solutions that their colleagues have found to particular translation problems.

Many translators dictate. But what they dictate varies: some dictate a very preliminary first draft which they then tidy up and revise on paper, others dictate an almost finished translation which only needs subsequent checking.

Others prefer to work on a screen, still others sometimes dictate, sometimes work on a screen, depending on the type of document or the urgency of the translation.

The motivations for using one method or the other are almost as many as there are translators. There are those who are convinced that dictation leads to better translation, either because the musicality of the language plays a greater role than when one types or because dictating leads to a simpler and clearer style, with shorter, more common, words and less complex sentence structure. Others dictate because their typing ability is poor, or because dictating is faster than typing. Even those who prefer to use a keyboard show a tendency to dictate when a piece of work is particularly urgent. Amongst those who prefer to use a keyboard, there are those who are technically minded and like using machines, those who always found dictation a painful and inefficient process because they frequently want to go back over passages that have already been translated, and those that were simply trained to use keyboards and find it hard to imagine doing anything else.

It was pointed out that choosing to use a PC rather than dictating can cause the translator to become involved in problems that otherwise fall onto the secretarial staff. He may, for example, spend time persuading a recalcitrant printer to behave as it should, or figuring out how to create and manipulate tables. There is also a connection here to the page counting issue mentioned earlier. If a translator types his own translation, he is saving secretarial work for his unit, but this is not directly reflected in his own productivity. Many translators are quite slow typists, and will therefore prefer to dictate in order to increase the number of pages produced, even if otherwise they might prefer to work on a screen.

Whatever the method chosen, it is important to be able to sustain a certain level of concentration. (Those who have been involved in the translation workshops talk of a "translation state" which takes time to enter and which is difficult to recapture once it is broken). Thus, interruptions are unwelcome, and distractions infuriating.

Distractions can come in many forms. Several people talked of the distraction caused by being in physically and ergonomically unsatisfactory environments: screens placed by windows, chairs and tables at the wrong height, construction work going on directly outside and so on. Others talked of the importance of what is on the screen: a flashing icon or a mail system pinging when a message arrives can break concentration and interrupt the flow.

For those who work on screens and perhaps use several tools at the same time the quality of the interface is also important. Interfaces must be integrated, and the same manipulation of the keyboard or mouse must always mean the same thing, no matter what tool is being used. This poses problems when commercial products have to be integrated into the computing environment. Computing environments must also be relatively stable. A translator cannot be expected to accustom himself to a new environment every few months. Furthermore, the capital investment involved in changing the computing environment is substantial. The SdT is heavily computerized, with more than twenty minicomputers, 1,200 PC's (which will become 1,500 by the end of 1995) and all the peripheral equipment such a set-up implies. Upgrading the environment also means adaptation of existing software. For all these reasons the computing environment in the SdT tends to change slightly more slowly than the state of the art. Thus, for example, although WordPerfect 6 is now available and is in many ways a superior Windows product to WordPerfect 5.2, migration to WordPerfect 6, or, possibly, some other word processor, will not happen until the end of 1995 or even later. Similar considerations determined the choice of DOS 5 rather than DOS 6.

The tools used by the translator during the process of translation are, as the above implies, those associated with document creation: the dictaphone or a text-processing system being the main ones. Many translators use spelling checkers to check their finished text. None use grammar checkers to check translations.

There was lively interest in voice dictation systems both amongst those who dictate and amongst those who type. These systems seemed to many translators to combine the best of both worlds. The translator would be freed from the pain of typing in his text, but nonetheless would have it available immediately on the screen, so that he could see what he had said, could scroll back to earlier in the text and so on. Many of those interviewed thought that the current limitations of voice dictation systems would not be a great drawback to their use in a translation service. That they are only capable of recognizing discrete speech is less likely to perturb people who are used to dictating, and that the system has to be trained for each voice that it has to recognize is not a great hindrance in a set-up where most people have a machine of their own. One group of translators were, though, very firmly opposed to the idea of giving voice dictation systems to the authors of the original documents.

It happens fairly frequently that a translator will find, whilst working on a text, that there are passages of the original which are ambiguous or which he does not understand. In this case, he needs to contact the author of the document to get clarification. This can prove problematic. The request slip asking for the translation contains the name of a contact person, but this will often be a secretary in the requesting service rather than the author. Once the author has been found, it can prove difficult to explain the nature of the problem to him. A common reaction that was quoted several times was for the author to say that the reader would understand anyway, without realising that the translator cannot translate what he himself does not understand.

An anecdote may illustrate this point and also provide some light relief. A recruitment test, drafted in French, contained the question "Qu'est-ce que c'est le GATT?" The multiple choice answer, amongst wrong answers, of course contained the correct "L'accord général sur les tarifs douaniers et le commerce". When the test was sent for translation, its author was astonished to be told that the question could not be used, since, once it was translated into English, the question contained its own answer and therefore favoured English speakers.

The most common way round this is to phrase the question as "So what you really mean is ..." and get an answer that way. Once again, the degree of exasperation aroused by the difficulty of contacting authors varies a lot from unit to unit and from department to department. And, of course, exasperation may not be one-sided: the requester who has been contacted by up to ten translators all asking the same question may well become a little impatient. The problem is that, in most cases, the individual translator has no way of being automatically informed that one of his colleagues has already asked for and obtained clarification. This is one of the problems that a bulletin board system might help to solve.

K.3.8.3 Other activities

A translator will frequently be involved in activities other than translation. Some of these we shall come back to in a later section, but an attempt will be made here to indicate the potential variety of the other activities which may occupy his time. It may be worth mentioning too that translators are not always unambiguous about their feelings to other activities. Of course, other activities add interest to the working day, but several who were heavily involved in them said that they sometimes felt guilty towards their colleagues, or felt that their colleagues resented their involvement. Others feared that they might damage their own chances of promotion by seeming less productive in terms of number of pages translated.

Each unit has one or two computing correspondents, often one translator and one secretary. The correspondents serve as liaison between the unit and the computing services, and help their colleagues with computing problems where necessary. In many units, it is primarily the secretary correspondent who does most of the work. But in those units where there is a translator, correspondent or not, who is familiar with computing and comfortable with computer tools, his colleagues tend to address themselves first to him, and he may well become very involved in this kind of work. The departments also have a computing correspondent, but he is not usually a translator.

A translator may also be a terminology correspondent, helping his colleagues with terminology problems, perhaps helping to create local terminology resources and liaising with the Terminology and Language Support Services Unit.

He may also be a training correspondent for his unit, with the job of keeping himself and his colleagues informed of training possibilities.

He may also be involved with the Permanent Delegation of Translators, which in normal times meets once a month and discusses questions related to management of the Service. The delegation fulfills, for example, a watchdog function on questions of modernisation and on how they affect the staff of the Service, and plays a highly visible role in providing feedback to the decision process and expressing the translators' point of view vis-à-vis the hierarchy.

Translators are very actively involved in the evaluation of new products which are being considered for integration into the SdT environment. At the end of 1994, three commercial translator's workbench systems were being evaluated and two local terminology data base management systems.

He may also be involved in the further development of existing tools. Thus, although an experiment on using SYSTRAN for the translation of texts dealing with nuclear matters and with policy on regional development is coordinated through AGL 3, translators are in close contact with the project.

Both translators and secretaries may also be involved in Steering Committees ("Comités de Pilotage") which are set up to follow particular projects. Each steering Committee is normally chaired by a Head of Department, and its members will include the Head of Project for the project in question. Additional members act as representatives of particular user groups, and an attempt is made to include every group which has a legitimate interest in the project. As a representative of his own constituency of users, each member is expected to consult his colleagues and to present their communal views rather than his own individual view. There is always a representative of the Permanent Delegation of Translators. There are currently four major projects, which each have a Steering Committee. The first is concerned with Follow-up, the redesign of SUIVI. The second is concerned with integration questions on the PCs, especially stream-lining the user-interface. The third deals with terminology issues, and the fourth with integrated translation tools. A fifth steering committee may soon be set up to follow work on the document server project.

K.3.9 In-house translation: the secretaries

A text which has been dictated is typed by a secretary. All secretaries use WordPerfect, which was initially selected for its ability to support multilingual character sets and maximize document interchange compatibility with the Community institutions. They also use the standard facilities provided with text-processors such as style-sheets, macros and spelling-checkers considerably more than do their translator colleagues.

Secretaries are also asked to type from handwritten input. This is usually a consequence of corrections to draft translations by the translators who have dictated their work, or of corrections made in the course of revision. Inevitably, there are people whose handwriting renders this task difficult.

A problem with keeping secretarial staff was frequently mentioned. The SdT is the most typical first destination for a newly recruited secretary, partly because of the high number of secretarial staff needed by the SdT and partly because of frequent departures. Working with the SdT does have some advantages, primarily those of working with fellow-nationals and of comparatively flexible working arrangements in some cases. Some people too, for a variety of reasons, are happy to have work which does not vary a great deal and which does not imply a heavy load of responsibility. However, audio and copy-typing is not a very interesting occupation, and career-oriented staff tend to leave for posts with more varied work as soon as they can.

It was suggested that secretarial staff could do part of the preparation of a text: in certain units, for example, secretarial staff already carry out preprocessing of repetitive texts so that what is passed to the translator is a text where standard repetitive phrases are already replaced by the target equivalent. This technique is being used with considerable success for some texts, of which the Bulletin is one noticeable example. The degree to which preprocessing can be used to lighten the burden of translation or of typing depends on the identification of suitable types of text.

Although one of the questions put during the interviews aimed at getting an impression of how much repetition those working on them perceived in the texts they dealt with, it proved impossible to formulate any clear opinion. This perhaps can be partially explained by the fact that repetition can occur at any level of a text. There are the obvious clear cases where large stretches of text are repeated, but, at the other extreme, there may be heavy repetition of quite small text elements, which is harder to spot. Also, there is quite a strong difference between literal repetition, where a large or small segment of text is repeated in exactly the same form, and what one might term intuitive repetition, where repeated segments are almost, but not quite the same. Literal repetition lends itself to computer processing in a very straightforward way, where intuitive repetition may imply recourse to more sophisticated techniques. Nonetheless, for the translator, both kinds of repetition feel very similar, and he may be disappointed if a tool which deals very well with text exhibiting a high level of literal repetition fails to perform equally well on texts with a high level of intuitive repetition. This in turn leads to the reflection that there may be a place for the intervention, at least in the early stages of trying to identify suitable texts, of advisers specializing in the processing of repetitive text (Language Help Desk) and software tools designed to confirm intuitive judgements of text repetition (for which TMan is, in effect, a pilot). The experience acquired through the new Translation Workshop (q.v.) may also help with this.

Some units make use of freelance typists. This can prolong the work-chain: there are cases where a text is sent out for initial typing, comes back for proof-reading, is sent out again for correction, comes back again for revision by the translator, goes out again, comes back for correction, goes back again to the freelance, and so on. Freelance typists return the final version of their work on diskettes, but do not always return early versions which are to be corrected in electronic form. One of those interviewed pointed out that this means that the time taken over the whole chain of typing and correcting can become unnecessarily long: if the typed version of the first draft were returned on diskette, the translator's and reviser's corrections could be done directly on the electronic version and one or more steps in the chain avoided.

A number of those interviewed also remarked that the typed final version tended to arrive on the last minute, so that there was little or no time to do final corrections. This was found frustrating, especially since there was no guarantee that the freelance typist had succeeded in deciphering the translator's or reviser's handwriting correctly. Handwriting can sometimes be appalling, and when typing is done by a freelance, the author of the correction is not easily available for clarification.

Whether a translation is prepared directly in electronic form by the translator or whether it is typed into a text-processor by a secretary or freelance, the electronic file is periodically sent to a central archive, where it becomes available for consultation.

K.3.10 Revision

How much revision is done varies enormously from unit to unit. There are units where every single translation is revised. There are others where the resources to do this are not available. In these cases, the criteria used to decide what texts will be revised include the importance of the text itself, its length, whether it has been translated by a newly appointed translator, and so on. Some units try to send to freelances only texts where no quality control is required, in order to avoid having to revise freelance translations. Others try to revise all freelance work. Still others do not have the resources to revise all of it, and are obliged to rely on spot checks.

Policy varies as well about who will do the revision. In some units, revision is seen as part of an hierarchical structure, where a senior person revises the work of a more junior person. In others, revision is seen as team work: two translators sitting together on the grounds that four eyes see more than two.

The tools used by revisers are the same as those used by translators, except that a reviser will not normally dictate his modifications to the translation. He will either hand-write the corrections on the paper version or work directly on the electronic version. Otherwise he will make use of all the tools and resources for documentation and research that the translator himself uses. Indeed, there is a sense in which a reviser's need for quick and efficient access to documentation is even greater than that of the translator, in that the variety of different texts that the reviser may deal with in any given period of time is greater than that dealt with by the translator.

K.3.11 Return to the requester

Once translation is completed, the translation is returned by e-mail to the requester. We have already mentioned that some requesting services are not yet very familiar with e-mail, and that difficulties can be experienced at this point. This is obviously a problem that time and a change in work habits will cure.

A greater problem from the point of view of the translator is that of obtaining feedback. It is worth pointing out that the requester is not normally the end user of the translation, and indeed may not know the language the text has been translated into.

It has already been mentioned that some units cooperate with specialised organisations in the Member States in order to get feedback, and that some individual translators occasionally attend the meetings where their translations are used for the same purpose, but no general solution to the problem has yet been found.

It has been suggested that each translation should be accompanied by a form explicitly requesting feedback, but the suggestion has not been tried out and it is difficult to see how useful it would be in practice.

K.4 Support activities

In this section we describe the activities designed to lend support to the primary activity of translation.

K.4.1 Terminology and the terminology help desk

Easy access to good terminological resources is of critical importance to both translators and revisers. This has long been recognised by the SdT, which maintains a staff of terminologists. Their work can loosely be classified as proactive or reactive.

Proactive work involves identifying new terminology, updating and maintaining EURODICAUTOM, working towards greater collaboration with the other institutions and planning for the future.

EURODICAUTOM should, ideally, be a multilingual, multicultural terminology base, easily accessible through user friendly interfaces and covering all Union languages. Although widely acknowledged to be an invaluable tool, in its current state it does not meet those ideals. That this is so follows inevitably from its age: it is based on old technology, the number of languages it should cover has grown over time so that coverage of languages is unequal and a certain amount of indiscriminate up-dating has led to the existence of double or multiple entries for some terms. Nonetheless, it is available to all officials, and to the citizens of the Member States at no cost except the price of communication, is therefore very heavily used and highly competitive with commercial terminology bases. For the future, the challenge will be to modernize without loosing the invaluable information which has already been collected.

Most modern terminologists think in terms of concepts. Creating a new entry involves answering questions like whether the concept being dealt with is stable, whether it is the same across different disciplines and different environments and so on. Such questions are answered through background reading, through research in data bases and through reflection. Several terminologists pointed out that one way in which a terminologists's work differs from that of a translator is that the translator is always under time pressure: he has to find a solution to the problem posed by his current text. The terminologist can afford to search for a good solution, even if that takes some time. Amongst the tools used by the terminologists, the data base of press releases (RAPID) and the various corpora of newspapers were frequently mentioned as being very useful.

A term is accompanied not only by the equivalents in the other languages of the base, but also by a classification code which indicates the subject field in which the term is used. The terminologist is thus also presented with a problem of classification. Subject codes are, by their nature, open to interpretation, and since it is impossible to predict who the user of a terminology base may be, the terminologist has to develop what one of the people interviewed called "a sort of universalism" which will help to maximize the chances that the user's interpretation will match that of the terminologist.

Other information attached to a term includes managerial data which, for example, gives an indication of the age of the information and its validity, and encyclopaedic information which helps the user to identify the nature of the concept underlying the term.

The creation of new terminology is also contracted out to freelance terminologists. As with freelance translation, problems of quality have sometimes been encountered. This is an acknowledged danger of paying for terminology as piece-work.

Terminology is also obtained from other sources through collaboration agreements. Although material from other terminology bases is an obvious source for finding new terminology economically, it can give rise to consistency problems once incorporated into EURODICAUTOM.

Updating a terminology base involves inspecting entries to discover whether the information they contain is correct and is correctly presented. A term may, for example, fall into misuse or change in the way it is used. Here, the questions to be answered remain essentially similar to those involved in creating a new entry.

A special problem is caused by the existence of multiple entries for the same term. A natural desire to be as complete as possible has led sometimes to a new entry being added for a term for which an entry already exists. Notice, though, that this does not necessarily mean that the later entry is redundant. We have just mentioned that a term may change in usage over time, and we noted earlier that the same term may be used rather differently in different environments. Over long periods of time, though, the result can be that the user is presented with a great deal of noise, and has to sort through many entries to find the one that he really wants. Until quite recently, even identifying cases of multiple entry was difficult and time consuming, involving individual searches and a certain hit and miss factor on whether a multiple entry was found or not. The recent implementation of batch mode searching in EURODICAUTOM facilitates identification of multiple entries, and makes it possible to envisage some of the preparatory work for cleaning up the existing entries being done by freelances. Sometimes, when a multiple entry has been found, it is not a question of choosing what entries should remain and what should be eliminated, but of synthesizing two or more of the multiple entries to make a new single entry. This is reported to be difficult but fascinating work.

Reactive work in terminology essentially consists in trying to help the translator who is confronted with an immediate problem. The translator makes use of the SVP service, run by the terminology help desk, whose staff reply not only to linguistic questions but also to questions about the use of existing tools. The help desk staff become expert in the use of EURODICAUTOM, and point out that experience leads to discovering ways of searching, based for example on the subject codes, that the average translator does not know about but which lead to a higher success rate. The help desk staff suspected that translators sometimes did not consult them about how to use EURODICAUTOM because they felt that it was embarrassing to have to ask what they believed they should already know. Other translators, they felt, tried to search for themselves, but became discouraged if they did not find the information quickly, for example because of having used an ill-chosen subject code; hence the suggestion that creating short courses on "clever" ways to search EURODICAUTOM might break down some of the barriers to self-help.

Because the job of the help desk is to be on call in order to react to requests for help, it is clearly not possible to plan the work. There are periods of intense activity, but they are impossible to predict. Most of the help desk staff therefore participate heavily in other activities such as evaluating new products or helping to develop SYSTRAN. We shall return to these activities in section (K.4.4) and in section (K.4.7).

As well as the centralized terminology staff concentrated in AGL 3, each language unit has its own terminology correspondent. There are also local terminology bases scattered throughout the SdT, whose creation is sometimes the result of a concerted effort by a unit or a department, sometimes the initiative of an individual translator. Commercially available products for the management of such local data bases are currently under evaluation and we have already noticed the potential of such local bases as sources of terminology. However, several of the people interviewed pointed to the dangers inherent in having a proliferation of local terminology bases at the expense of the central resources, and to the problems of validation associated with sharing which have already been mentioned. Indeed, one of the people interviewed characterized the major problem of terminology management not as a problem of planning but as a problem of how to share knowledge.

K.4.2 Documentation

The importance of documentation has already been underlined several times, but it is probably worth emphasizing the critical role it plays in translation once again here, where we are directly concerned with the provision of documentation. We have already seen that producing a translation involves having acces to reference documents which can be numerous, are not always supplied by the requesting service and can be hard to find. Apart from direct reference materials, translators and terminologists frequently require other documentation during the research phase of doing a translation in order to establish correct terminology, to check usage, to consult previous translations or to find citations.

With the growth of the European Union and of the information society, the mass of documentation produced becomes ever greater, and the task of finding the relevant documentation becomes ever more complex. The role of specialist documentalist services subsequently becomes even more important.

The services of documentalists are available through the documentation centres (one per language in each of Brussels and Luxembourg), as well as through the Terminology unit's Reference Library (AGL 3). The terminology centres, although spatially separate, are coordinated by AGL 1.

The work of the documentalists focuses on the management of documentary resources, the collection of new resources and research in the resources available and in other data bases to find the solution to specific problems.

The documents available include standard works – monolingual, bilingual or multilingual – originating from outside the Commission, such as monographs, dictionaries, encyclopaedias, yearbooks, updates etc. Some CD-Roms have been acquired recently. They also include works originating from the Commission itself: the Official Journal of the European Communities, preparatory documents (COM, SEC, ...), reports (statistical and general), Bulletin of the European Union and so on. The official Commission documents are available in all Union languages.

Materials are collected either through direct purchase or acquired from the Commission and other institutions.

Finally, scientific, technical and linguistic reviews originating from outside the Commission are obtained by subscription. Not all languages can be equally covered. (There is a strong and growing tendency, for example, for the scientific community to publish a great deal in English, and very little in, say, Danish or Greek.)

The specialist magazines are distributed to both the translators and the terminologists, and the relevant articles are indexed in Terminology's local documentary database, TRADOC, which contains references to Terminology's stock of books and articles. Some AGL 1 documentation centres have a separate documentary system or database covering their own specific needs.

Research is done in response to requests from translators and terminologists, and typically involves searching the Community data bases. The latter include

CELEX the database of legislation, court judgements and preparatory documents. CELEX contains full text in all the languages of the Treaties, the regulations, the directives and some other documents. Entries not in full text have bibiographical references.

ACTU a database containing bibliographic references as well as administrative information (for example, the D.G. responsible) relating to the most recent Commission documents. ACTU exists only in French.

EC01 a documentary database containing the full text or at least a summary of minutes, Commission decisions, Parliamentary questions etc. EC01 also exists only in French.

RAPID an online database providing rapid access to the full text of all documents issued by the European Union's Spokesman's Service (for example, press releases and speeches). RAPID covers all the languages.

ECLAS (**E**uropean **C**ommission's **L**ibrary **S**ystem), a bibliographical database of the Commissions' Central Library (D.G. X).

In the course of their work, documentalists often need to contact requesting departments or national bodies, for example ministries, geographical institutes etc., in order to conduct their researches.

The documentalists also make use of data bases external to the Commission services, and would like access to more still. Several people, translators and terminologists, pointed out that one problem in consulting a wide range of different data bases was that the means of access and of query varied from data base to data base. The help of documentalists and others who had acquired specialist knowledge in this respect was much appreciated.

The clientèle of the documentation services is very wide. As well as translators and revisers, requests from officials outside the SdT, from interpreters and from freelance translators are also received. This gives rise to a growing problem; more and more interpreters and freelances address their queries directly to the documentation services, taking it for granted that the service will also undertake documentary research and supply glossaries, on demand. The service cannot meet all the requests made, and there seems to be no clear policy about which requests they should try to meet. Even, on occasion, the service can be unsure about whether it has the right to give out some of the material requested. A document distributed to all staff, "Guide d'accès aux documents de la documentation", sets out guidelines on confidentiality issues and the procedures for third party access to Commission documents, but only covers the procedural aspects of giving access to full documents. Some practical guidelines tailored to SdT needs and concerning the distribution of excerpts from documents, glossaries etc., might prove useful here.

As one of the people interviewed put it, the documentalists are at the centre of the information revolution. Ways of work are changing rapidly, and with them expectations of what can be provided in what form, but new ways of working have not yet settled into their proper place.

K.4.3 Training in computer tools

The SdT is heavily computerized, as we have already noticed. This implies considerable training needs. In particular, the migration from the Unix based architecture to the new PC-based client server architecture created special training needs, amounting to more than 6'000 student/days. This large training effort started at the beginning of 1993 and will not finish until the end of 1995. It constitutes a heavy burden for the language Units, who have had to call on colleagues in other departments and sometimes on outside help.

An effort of this size is exceptional, but it should be noted that any new software which is installed brings with it a training requirement. The evaluation of new tools also brings with it smaller scale one-off training requirements for those who will be involved in the evaluation.

Workshops on the use of office aids are organised regularly, and the SdT organises its own training courses in the use of CELEX. Courses on project management are aimed specifically at the computer staff and at project leaders.

During 1994 six half-day seminars on new tools and their use, including the possibilities offered by the new computing environment, were organised for the Heads of Unit and their deputies. More advanced seminars on the use of some of the new tools will be organised in 1995.

K.4.4 The computing correspondents

The staff of the computing services (SdT/02) are distributed over Luxembourg and Brussels. The activities of the unit fall into two broad categories, training, support and help with the use of computer tools, and development.

In 1992 it was decided to decentralize the support and training activities, situating them directly in the departments and in the AGL directorate. The decentralisation was based on three levels of organisation, full-time computing correspondents, part-time computing correspondents and a central computing help desk. This structure became fully operational at the beginning of 1994.

The full-time computing correspondents are an integral part of the central help desk. They participate in its weekly meetings and in the day-to-day staffing of the help desk. Operational problems are discussed in a meeting between the full-time correspondents and the part-time correspondents from the language units which takes place every six months.

In November of 1994, it was decided to take decentralisation a step further, with the central help desk in Brussels being abolished in favour of five departmental help desks, reporting directly to the Head of Department but acting in coordination with SdT-02. Each help desk has two full time staff. The new structure in Brussels became operational at the beginning of 1995.

As familiarity with the new environment grows, the role of the help desk also expands beyond simply helping the users to adjust to the new environment. The enlarged responsibilities of the help desk cover

- Help to users, both in preventing problems and in solving them
- Training of users
- Information to users
- Feedback on users' needs to the computer services
- Beta-testing of software
- Orientation of users and standardisation of work methods
- Advice and support to users who are carrying out their own developments.

The staff of the help desk, on top of their own expertise, need to know who to go to when unknown problems present themselves, and how to encourage the individual users towards finding the best method of working with the tools that are offered to them.

K.4.5 Evaluating translator's work bench systems

Three commercially available translators' work bench systems (TWBs) were under evaluation at the time of the interviews, with translators from the language units directly involved. For most, the main feature of the TWB is perceived to be the translation memory function, although one translator reported having used the term recognizer of one system with very satisfactory results: in the text he was translating there were three terms where he could not find help even from a specialist he consulted. With the term finder he got the terms and context very quickly and from a source he could trust, the Bulletin. Another reported using a TWB in order to create glossaries for the documents he was working on. The document was sent from the PC to the server which preprocessed the document and returned it together with the glossaries. Then, whilst he was working, the system proposed any solutions found during the preprocessing.

Testing the translation memory function relies on having something in the memory. One of those doing the evaluation had created a number of small memories, mainly representing text which had already been rationalised, another had simply taken the originals and translations of a repetitive text that arrives for translation every month and used those to stock the memory. Yet another had taken previous texts and their translations, without trying to select text that was thought specially suitable for machine treatment, for example because it was repetitive or followed a standardized format. The importance of what is in the memory was emphasized by the person who reported a very positive experience with a very long text which came back to him for modification: with the old version in the translation memory he had been able to complete the job in an afternoon, whilst his unfortunate colleague without a translation memory took a week for the same job. Several of those involved in evaluating translation memories said that it would be useful to be able to edit the translation memory, which is not possible with some commercial systems.

Most of the comments offered related to the interfaces of the system, rather than directly to functionality or efficiency, although several people said that they strongly disliked having to wait whilst processing happened. One person talked about what he called the "motorway syndrome": moving at all feels better than sitting waiting, so even though one might be aware that sitting in the queue is likely to get one to the destination quicker than leaving the motorway and tackling country roads, one still opts for the country roads. In a rather similar way, one is very irritated if EURODICAUTOM offers twenty-five solutions and it takes time to choose between them, even if doing so actually reduces the total time required to find a solution, or if a translation memory takes a perceptible time to come up with a proposal, even if the proposal fits exactly.

It is important to notice that two interfaces are involved, one for the user of the TWB, one for the management of linguistic resources such as the translation memory. Often the user interface was thought more satisfactory than the resources management interface.

The number of comments relating to the screen indicate how important presentation is to most people. There was much discussion of whether translation and original should appear on the screen together, whether the translator should have to over-type the original, whether it was satisfactory to have only one sentence at a time appearing in the translation window and so on. Several people remarked that it was important for the user to be able to organize the screen in the way he wanted. The number and size of windows was important, as was the ability to work without being distracted. A critical feature was that the focus of the screen should be on the translation being done. It was agreed too that it should be very easy for the user to use the system: for example it should be integrated with the text-processor the user was familiar with rather than involve use of a special system.

There was common agreement too that it would be useful if the unit of treatment were shorter than a whole sentence.

There was much less general agreement on other features. There were those who only wanted text and translation to go into memory when the translator decided to put it there, others who wanted automatic recording so that colleagues could have almost immediate access to what had been done. There were those who wanted search for similar texts and their translations only to be done when the translator requested, others who wanted it done automatically all the time. Many of these questions relate to the general issue of validation in an environment where translation memories are to be shared, an issue which has already been discussed in an earlier section. Several people suggested that it would be useful to work in batch mode rather than interactively.

Reactions to using an alignment facility varied enormously, and seemed to have a great deal to do with the particularities of specific products. Therefore, apart from reporting the obvious and general desire that alignments when presented should be reliable, this question will not be further discussed here.

The stage of the evaluation described here is now over, and final reports on each of the three products were presented at the end of June 1995. The evaluation will now continue in a real production environment in the framework of the translation workshop and modernisation network already mentioned, both of which are due to start in the autumn of 1995. It is felt that, although the first round of evaluation did provide a very good overview of what was available and the current state of the art, it did not allow for what is sometimes called in the evaluation literature "scenario testing" (see (B.4.1), testing of a product by putting it into use to solve real problems in a realistic work environment.

The workshop will be staffed by volunteers from the translation units and from the horizontal services, who, although they will remain attached formally to their own groups, will be temporarily relocated in thirty or so offices in Brussels. The workshop will acquire an expertise in using computerized tools to help in the task of translating documents identified as suitable material for such treatment. Liaison with the Heads of Department, Central Planning and the translation units will be done conjointly by the Director of AGL and the Head of Department A. When the staff of the workshop return to their own group, the savoir faire they have acquired will be transmitted to their colleagues.

The network is a parallel operation in Luxembourg, whereby the participants stay in their own Units, and a loose organisational structure coordinated by a member of the AGL staff links them together. With the aid of the coordinator, suitable documents are identified and are treated with new tools, primarily pre-processing tools and translation memories. The aim here is to investigate how well new tools can be integrated into existing structures and the need to create new structures avoided.

K.4.6 Evaluating local terminology management systems

Many of the comments made in discussing the evaluation of translators' work benches apply equally in the context of evaluating local terminology management systems, especially the questions of validation and the emphasis on good interfaces. It was pointed out in particular that the interfaces for terminology management systems need to be quite sophisticated because of the number of languages to be dealt with.

Local terminology management systems were essentially being used in two ways in the evaluation.

First, they were used as an intermediary between the terminologist or translator and the central resources of EURODICAUTOM. When a text was received, the user would search for the appropriate terminology in the central resources and down-load it into the local system. This was done with each document as it was received: the local base was not used all the time. The possibility of sending a list to EURODICAUTOM for search was said to be very useful in this context.

Secondly, local management systems were used as a way to create local terminology, for example requester oriented terminology, which was of special interest to the unit but not necessarily of central interest. Sometimes the terminology correspondent of the group was responsible for coordinating this work and perhaps for entering material in batch mode after validation within the local group. In this application, the group suggested that it would be useful if the requesting service could also have access to the local terminology base, especially if the terms were integrated into a document such as a working programme where it was important that terminology be consistent.

The importance of integration with other tools was also emphasized in this context, as was the need to be able to use the local management tool without being distracted whilst translating.

Two worries were expressed by those involved in the evaluation. The first was that the system being called a local data base management system might lead to unrealistic expectations on the part of some users, who would expect to find a system complete with data rather than a software skeleton onto which the data had to be grafted. The same concern surfaced also with the testing of translation memories: users' expectations can be unrealistically high simply because they do not differentiate very well between an interface and the data that lies behind it.

The second, already mentioned and very widely expressed, was that the availability of local systems would lead to a proliferation of scattered small terminology bases - a computerized equivalent of the time when every translator had his own private collection of translation equivalents, the contents of which were unknown to his colleagues and therefore not easily shareable, with the result that much work was done several times over and consistency across translators was much harder to achieve than it is when a central resource creates a de facto standard for terminology.

K.4.7 Developing machine translation systems

We have already noticed that the SYSTRAN machine translation system is available both to officials outside the SdT and to the SdT staff themselves. One experiment in developing the use of SYSTRAN further is undertaken with the collaboration of the staff of the language help desk (AGL 3). The two areas concerned are the nuclear sector and regional development policy, where the D.G. concerned already uses SYSTRAN without going through the SdT. In practice, the D.G. helps to build special glossaries for these two areas. The help desk staff then receive texts and send them for SYSTRAN translation, do the post-editing and extend the vocabulary where necessary in collaboration with the D.G.

Translators in the language units also help with development of SYSTRAN. One such experiment concerns anti-dumping legislation. The translators pick up the original text through the computer network, ask for a SYSTRAN translation, then provide feedback to the SYSTRAN development team by sending a copy of the SYSTRAN raw translation, their own translation and notes. The feedback is not organised through standardized bug reports, and the translators are free to choose the problems they will concentrate on. The translators involved in this kind of work found it satisfying to see the results of their feedback appearing in the form of improved quality of the SYSTRAN output.

Despite all this, there were still some reservations expressed about the use of machine translation. The fact that SYSTRAN is used outside the SdT with little control is not an unmixed blessing in this respect. On occasion poor quality unedited raw machine machine translations have even been published in the Official Journal. In these cases, as in the case of poor quality translations done in the requesting services, it is the SdT who gets the blame in the eyes of the outside world, and this is, quite naturally, resented.

K.4.8 Computing support, modernisation, development of multi-lingual tools

Much has already been said about the computing services and the support work undertaken by the AGL Directorate. In this section we summarize briefly the roles of those units most directly concerned with the specification, choice, implementation or integration of new computing tools.

Let us start with SdT-03, the small unit responsible for modernisation of work methods. As its name implies, it is the job of SdT-03 to look at the organisation of the work of SdT as a whole, analysing both the activity of the Translation Service and the administrative, organisational and technological environments within which this activity takes place, and to follow how these environments are evolving and changing. The unit also keeps an eye on what is happening elsewhere, in other organisations and institutions. On this basis, the unit tries to work out what the possible options are for improvements throughout the SdT, and to encourage a favourable attitude towards change and innovation.

What this boils down to is trying to rethink radically the whole process of translation, seeing it as in symbiosis with computer tools. Critical questions concern the relationship with the requestors and questions of how resources can be shared. This latter can be restated as trying to download as much as possible of the translator's know-how into a form that can be shared by others. Here, it is important to distinguish between recurring problems, where the problem and its solution can simply be recorded for future use, and more complex problems which are perhaps the domain of the specialist. The former involves above all trying to improve the communication channels, so that information does not get lost by being passed through a long and complex channel.

AGL 4 is responsible for the development of multilingual tools. In collaboration with AGL 3 (see below) and in consultation with SdT-03, AGL 4 studies what translators' aids tools are needed within the SdT, draws up functional specifications for preprocessing tools, authoring aids and standardisation tools, as well as for the terminology data bases. In collaboration with D.G. XIII, AGL 4 contributes to the development of machine translation systems in conformity with user needs. The Unit is also responsible for the choice of computer tools and linguistic resources, and for following their integration into the SdT environment. These responsibilities imply a need to keep abreast of developments in the language industry and in research. In the latter context, AGL 4 coordinates the SdT's participation in the language industry related R&D programmes managed by D.G. XIII and application of the results obtained by such programmes.

AGL 4 works very closely with SdT-02, the computing services, who as we have seen, are responsible for all the information technology in the SdT (including faxes and photocopiers, although not telephones). The job can be summarized as keeping the machines and the users operational, through maintenance, help desks and training. SdT-02 is also involved in the choice of new commercial products to be tested, their testing and their integration into the computing environment of the SdT. One problem repeated here was mentioned frequently by many of the AGL and computer staff: many, if not most, of the translators' aids tools available on the market are aimed at the single user. Adapting them to use in a very large service where information needs to be shared and double work avoided can be far from straightforward. We have already seen some examples of this problem in discussion of, for example, the use of translation memories. A second problem is that of integration, which is important both from the point of view of sharing resources and of user-friendliness. The user will typically have a work station with access to office tools, management tools and translation tools. It is important that these are all integrated in such a way that the same manipulation, typing "exit" for example, always means the same thing. This is a huge job, since different products are developed independently, and each manufacturer will at best strive for compatibility only between his own products.

Market forces also have an effect, sometimes undesirable, on the effort to keep all languages equal from the point of view of computer treatment. Text processors provide a good example here. English text can be dealt with through an English interface, and French through a French interface, but Finnish and Greek are typically dealt with through an English interface.

A PC-based client-server architecture also creates some tensions: an increasingly computer literate population of users is aware that almost anything can be done on a PC. But in a large organisation where resources have to be shared, some constraints have to be accepted if sharing is to be effective. It is not always easy to give the users what they want and still respect the constraints that make best use of the resources available, whilst still at the same time taking into account that another part of the targeted population is not at all computer literate.

We have already noticed that the size of the organisation has effects on the speed of change that can be accepted. In the interests of preserving a certain stability that will allow the user to be maximally productive, computerization in the SdT must always be slightly behind the state of the art whilst remaining keenly aware of likely future developments. This too is a delicate balance to maintain.

AGL 3 is the final link in the chain between the support services and the users. Its function is to make available a coherent set of language services and computerized tools to the translators and to the other officials of the Commission. Thus, the Unit provides a linguistic help desk, which can offer help with terminology and other linguistic problems, manages and develops the linguistic resources provided through central and local terminology bases, and helps to create new linguistic resources which will be integrated into translators' aids. The Unit is responsible for encouraging the use of language engineering tools and for their distribution, as well as for collaboration with the other institutions on harmonisation and exchange of terminology.

Before leaving this section we should mention the work of the Modernisation Committee and its successor, the Follow-up Committee, to whose work the groups mentioned in this section contributed very largely. The modernisation Committee was set up in 1993 and decided on a number of projects, including the creation of manuals for the data bases available, the introduction of e-mail throughout the Translation Service, the creation of topical glossaries and a project on rationalisation issues. The Follow-up Committee was created to follow the work of these projects. In the summer of 1995 it was felt that the aims of most of these projects were either achieved or well on their way to achievement, and that it was appropriate to make an assessment of the results of each action, and of where the results could be found. This has led to an information effort, in the form of a series of two to four page long documents (on yellow paper, in order to draw attention), under the name of "Mini-bilan", which attempt to summarise the results fo each action.

Currently, a slimmed down version of the Follow-up Committtee is looking at future modernization issues. The aim here is not to redesign the future shape of the SdT, but to look at practical measures like the introduction of POETRY into the requester services, starting with one Department and trying to ensure that every document coming in comes through PEOTRY, or trying to find alternative ways of easing the problem of information transfer which has been mentioned many times in this study and is widely acknowledged to be a difficult issue.

K.5 Dream tools

During the interviews many ideas for useful tools were suggested, either in the course of the conversation or in response to specific prompting. In a few happy cases, the tools already existed, and it was possible to pass on information about them either directly or by asking the SdT coordinators of the study for help. Great efforts are made within the service to ensure that information about what is available circulates freely, but the sheer size of the SdT, coupled with the complexity of its organisation and the heavy workload inevitably means that sometimes information does not get through to the person who needs it. It was pleasing to be able to offer useful information from time to time.

Some of the tools suggested were already being studied by the computing services or by AGL 4. Others, although they may not yet exist on the market, are perfectly feasible in the state of current technology. Still others would require a research effort for their development, but are imaginable in the current state of the art. Yet others are tools for the future: they would require major breakthroughs to find solutions to problems that until now have proved intractable.

Some of the ideas are not tools at all, but changes which would facilitate the work of the SdT (or in one case, eliminate it!).

It is perhaps worth noting that willingness to suggest new tools was strongly associated with an awareness of the usefulness of those tools which exist already.

The ideas suggested are simply collected together here, with no attempt to say what already exists, what is feasible and what is not.

- A single language for Europe.
- An ergonomic advice sheet: on this would be recorded all the data needed for an individual to create a work environment tailored to his physical needs: e.g., required height of chair, slope of chair back, height of table, position of screen, etc.
- A standardized complete modernization of administrative language.

- Full standardisation of texts.
- A standard observed even for relatively minor features of a text, like how footnotes are treated.
- Easy access to information available on CD-Roms, e.g. corpora, specialized dictionaries, encyclopaedias.
- Easy access to any data base which exists.
- Make sure that officials in the requesting departments also have access to terminology, to background documents and to reference documents.
- Persuade Council to accept electronic documents.
- Allow teleworking from anywhere in the Union.
- All staff completely networked.
- Tools that would facilitate translators working together in a group.
- Access for freelance translators to all the resources available to in-house translators.
- A scheduling programme which works out when a job should be started so that it can be finished on time, and does it dynamically to take account of new work coming in. Each day it would signal what should be done that day.
- A tool that would allow profiles of individual users to be set up, so that interfaces could reflect the individual's own interests and what he uses.
- A multilingual spelling checker which automatically recognizes what language is being dealt with and switches to the appropriate spelling checker for that language.
- A spelling checker that learns what mistakes the particular user makes and corrects them automatically.
- A tool that would check the quality of a freelance translation and set a warning flag if the translation should be thoroughly checked. It was suggested that using existing spelling and grammar checkers might be a point of departure for such a tool. It was also suggested that looking for characteristic translation errors might have potential.
- A style checker tailored to Commission jargon and style.
- A grammar checker that could learn a house style.
- An editing pool which would take documents on their arrival in the SdT, would sort out any problems in the original and would do as much preparation as possible before the document was passed onwards for translation.
- A multilingual authoring tool. The user would construct his text by putting together predefined textual elements chosen from a list of such elements. The system would generate equivalent texts in all other Union languages.
- Voice recognition for computer commands.
- Voice recognition for e-mail addresses, thus avoiding mistypings of non-natural strings.
- Good voice dictation systems for all languages
- A system that would associate spoken target language templates with words in the original text. This would facilitate dictation.

- A tool combining speech input and conventional text-processing: when revising, for example, one could use the mouse to highlight the section to be replaced and then dictate the modification.
- A system that would take a text, discover what all the relevant reference material was, generate that material and make it available.
- Inter-Institutional collaboration on reference materials.
- A system that would extract new terms and identify their equivalences.
- A tool that would check for the consistency of terminology across a text or series of texts.
- A dedicated computer scientist as part of the terminology staff.
- A tool that would identify all the repetitious elements in a text, and maybe identify what texts are suitable for preprocessing.
- A tool that would be able to recognize recycled text, even when there are changes in the order of text elements or minor changes of vocabulary.
- A tool which would examine a text and give advice on what other tools could be used on it, for example Tman preprocessing, machine translation, term extraction, translation memory.
- A translator's workbench with a translation memory which would find all terms for which a validated translation exists.
- A hierarchically organised terminology search system, giving preference first to validated terms, then to EURODICAUTOM terms, then to terms recovered from translation archives.
- A data base of external experts who can help with terminology, what their specialities are and how to contact them.
- A translation memory which would find all the old translations relevant to the current translation. Where the individual user had a preference for a particular translation, this would be signalled and used.
- Training courses for new staff to teach them how to use SYSTRAN in practice.
- Machine translation even of inferior quality from the lesser-used languages into English and French.
- A tool which integrates the best possible translation memory system with the best possible machine translation system and is then used as an aid to human translation.
- Fully automatic high quality translation.
- Machine translation with voice input.
- A CD-Rom jukebox giving access to all COM, SEC and C documents.
- A Commission-wide fully integrated document management and archiving system. The ideal version of this tool would include:
  - Every document and every translation ever produced, whether on paper or electronically
  - cross-indexing to show relationships between documents
  - all documents multilingually aligned
  - facilities for keyword search
  - facilities for content-based search

  - terminology resources, including full information on each term, such as
    * who uses it
    * how often
    * what the reference is
    * grammatical information
    * "views" to allow human use or computer use of the same information

- A project to think about all the different ways that massive amounts of linguistic material could be searched and used.
- An electronic thumb-print for every document so that it could be uniquely identified for ever.
- The ability to search for documents with a similar thumb-print to the one you are immediately dealing with.
- A tool to carry out consolidation of legislation which has been modified by other legislation, modifying each change in each text.
- An eye-pointing device to position the cursor on the screen.

K.6 Conclusion

This concluding section takes the form of a commented summary of the contents of the report. Each section will be summarized in turn, and an attempt made to pick out the main points which emerged from that section. Almost inevitably, those points will often touch on what the participants saw as problem areas. Repeating them here should not be misconstrued as hostile criticism.

- The first section was concerned only with preliminaries, describing the genesis of the study and the methodology adopted. Apart from reminding the reader that the study was based on a series of very loosely structured interviews with forty six staff members of the SdT, covering all the hierarchical levels, there is no need to recapitulate it here. It is however perhaps worth repeating that no claim is made to objective validity or to a statistically significant methodology.
- The second main section was concerned with background.

  We noticed that the SdT was responsible for dealing with an enormous volume of text, producing more than a million pages of translation a year. We also noticed, both here and elsewhere, that the documents sent for translation are very frequently the product of many different authors, and their authors are often not writing in their mother tongue. Consequently, translators are called upon to a degree unknown in other translation services to deal with language which is not of native speaker quality.

  We noticed too that because of the equal status of Union languages, the term "translation" was only meaningful as long as a document remained inside the Commission. Once it left the Commission there was no indication of what the original language of a document might have been, and all language versions enjoyed equal status.

  The SdT is a very large and very complex organisation, with around 1,700 staff organised into departments according to requesting services, into Units according to language, and into horizontal groupings providing a variety of support functions. In addition to the in-house staff, heavy use is made of freelance translation. If we also take into consideration the relations with the requesting services, it is hardly surprising that two topics, information flow and work flow, surfaced very frequently throughout the study, with work flow, both inside and outside the SdT, in particular, often being perceived as a major source of difficulties.

The SdT is heavily computerized, and many computer based tools are already available as aids to management and as aids to translation. New projects are continually being defined and developed, and consideration of future plans took us into some of the problem areas for which solutions are sought.

Many of these problems are the almost inevitable result of an organisation which is continually growing in size, and which is also having to face the challenge of moving from paper-based information processing to electronic treatment. Thus, the major problem of document identification (the inter-relationships between the SdT internal document identification code, the COMM, SECC and C document numbering systems) can be seen both as the result of historical process and as a problem calling for urgent resolution in the light of new possibilities offered by electronic archiving.

In term of solutions to particular problem areas, it was particularly noticeable that a pragmatic attitude had already led to ingenious and innovative solutions. As examples, we might cite the use of AVIMA to generate multilingual versions of calls for tenders, thus relieving the SdT of the necessity to produce a second million pages of translated text every year, the development of SEI-BUD to deal with the production of the budget in the form of a single integrated document which could be worked on simultaneously by all of the three main actors in its production, and the use of TMan to replace repetitive text in the monthly Bulletin of Activities, cutting translation needs by about half.

On a more general level, we noticed the growing use of the SYSTRAN machine translation system by officials outside the SdT to obtain rough translations without passing through the translation services, and the use of SYSTRAN within the SdT to get rough translation of background documents or, on a more limited scale, as a starting point for more polished translation, as well as a means of obtaining terminology or CELEX references.

Amongst the more recent products being evaluated for integration into the SdT's array of translators' aids tools, translators' workbench systems and local terminology management systems were prominent. A lively interest in voice dictation systems and the possibilities they might offer was very widespread, both amongst the translating staff and amongst the staff from the horizontal units.

On the organisational side, we noticed that a number of schemes for Inter-Institutional collaboration were under-way or being planned, ranging from quite modest attempts to collaborate on training and recruitment to more ambitious projects such as a single terminology base for the European Union, or to resolve the document identification problem.

- From an attempt to sketch in the background, we turned in the third main section to an examination of the document production chain and the work of those who participate in it.

  Starting once more from the original documents sent for translation, we noticed that the demand is not only large, it is also very variable, and that, large as it is, it does not reflect the total amount of translation done within the Commission. As well as the use of SYSTRAN already mentioned, it is not unusual for officials to translate themselves, or to arrange for translation to be done without going through the SdT. We also saw that this can lead to problems, since the SdT is perceived by the outside world as being responsible for the quality of **all** translation, without distinction, and, in any case, the outside world normally has no way of knowing whether a text was produced by the SdT, by the requesting service or by SYSTRAN. Translators sometimes understandably resent poor translation appearing in official documents.

  Most documents are still transmitted on paper. Strenuous efforts are in hand to increase the number of electronic submissions, and all translations are returned in electronic form, no matter how they have been submitted. Using e-mail for document transmission is not yet part of the normal way of life in the requesting services, although it is gradually becoming more familiar, and problems like a requester not knowing how to print out his urgent translation are becoming rarer. It should be noted though that a considerable proportion of documents originate from outside the Commission itself, from the Member States or elsewhere. In these cases, whether or not a document is transmitted in electronic form may not be within the control of the requesting service.

The requesting services do not always use the same text-processing system as does the SdT. Conversion programs exist, and the use of a standard for document preparation, EUROLOOK, is being promulgated not only in order to ensure a uniform appearance of Commission documents but also in order to guarantee convertibility.

Apart from these banal, but nonetheless important questions, a number of issues surfaced here that have to do with engineering the work flow between the SdT and the requesters, and, to some extent, within the SdT itself. Many of these have to do with two problems which were raised over and over again, the problem of version control and the problem of acquiring all appropriate reference material for a translation. Others, perhaps even more intractable, had to do with the difficulties of planning, especially in certain departments where the workload is extremely unpredictable and urgent work coming in can have serious consequences on planning that has already been done. This issue is complicated by the fact that there is no central policy on how to resolve conflicts. Yet other work flow problems have to do with at what point in the document production chain certain operations on a text might best intervene. For example, if there are problems with the source text, can they be trapped already when the document arrives in the department and resolved before the document is passed on down the chain to the translation unit or even the individual translator? Could some work on terminology be done high in the chain, and thus avoid reduplication of work at the individual translator level? Where should any preprocessing be done? Some of these questions also have a bearing on the issue of relations with the requesting services, who will obviously prefer to be asked a question once than to be asked the same question by a number of different translators who are all working on the same document.

Not surprisingly, information flow and communications issues also appeared largely in discussing the work of the Department Heads, as did the use of management tools like SUIVI. The question of encouraging standardization of documents also came up again here. Apart from the need to be in constant communication with the requesting services, the most commonly cited difficulty was that of trying to ensure that translation into the different languages proceeded in parallel. A new issue that came up in discussing the work of the Department Heads was the possibility of distinguishing different kinds of translation for different purposes: of suggesting to a requester, for example, that a quick oral summary might suffice for his needs, or that he might consider asking for a SYSTRAN translation. One striking aspect of the work of the Department Heads is the rich variety of activities and responsibilities they undertake in addition to direct task of liaising with the requesters and seeing that the translations get done.

Quite a large proportion (around 25%) of translation work is dealt with by freelance translation. A number of participants pointed out that it was difficult to determine the economics of using freelance translation, since direct comparison of costs makes little sense, but that, on the other hand, since it is simply not feasible to maintain a full-time permanent staff capable of dealing with the peak load, use of freelance work was inevitable. Freelance work is also used to deal with specialized texts, and we noticed that there is a long term policy to identify families of documents that could systematically be sent for freelance translation, thus easing the general work-load. When the interviews on which this study is based were beginning, a change in the procedure for the selection of freelance translators had recently come into effect, and a number of those interviewed expressed worries about how the change would affect the quality of freelance work. Given the very large numbers of people involved, of language combinations and of subject areas, no really thorough testing of those who had proposed themselves had been possible, and freelance translators were in effect being field tested by being used. It will take some time before a well founded judgment of the effects in the change of procedure can be formulated.

Use of freelance translators once again raises issues of communication and of work-flow, as well as specific issues of access to the various facilities and support services available to the in-house translator.

Another way of easing the burden on the translation services is to make use of machine translation. We noticed that the Commission has had a long standing interest in this area, and that the SYSTRAN machine translation system is now quite widely (and increasingly) used by officials outside the Translation Service. The SdT remains however the largest single user of SYSTRAN, and translators use it to obtain rough translation of background documents, to obtain CELEX material and other reference material and as a way of accessing terminology as well as sometimes to obtain a raw translation as a basis for more polished work.

A document which follows the conventional track through the SdT will go from the Department planning service to the Translation Units, which are organised around the individual languages. The Unit Heads, whose job is to make sure that translation of appropriate quality into their language gets done on time, either within the Unit or through freelances, emphasized the need for local management tools. They also stressed the utility of standardization in document preparation. Since the Unit is responsible for the quality of all translation done within the Unit, it is perhaps not surprising that several of the Unit Heads raised the question of quality control of freelance work. Quite naturally, the problem was felt most acutely in those Units where there was not the capacity to do much revision of freelance work. The Unit Heads carry quite a lot of administrative responsibility in addition to work directly related to seeing to it that translations are produced. Their vulnerability to interruptions frequently meant that they limited their own translation work to short translations or to very urgent work, although all seemed to be heavily involved in revision.

The document will find its way, eventually, to the desk of the individual translator, who may have been allotted that particular document or who may have chosen it out of the work waiting to be done, depending on the policy in his particular unit. His first task is one of research: of finding the reference materials, the terminology and the citations needed to complete this particular translation. Typically he will make heavy use of EURODICAUTOM and/or of CELEX at this point. These two aids to translation are used far more heavily than any others. EURODICAUTOM, in particular, was frequently cited as the absolutely essential tool. This is not to imply that both translators and terminologists were unaware of the weaknesses of EURODICAUTOM, many of which are inevitable consequences of its age and of its size. Creating a replacement for EURODICAUTOM or even simply bringing it up to date and cleaning it up is a mammoth task which also presents the challenge of not forfeiting the immense riches already contained therein.

A potentially rich source of future terminology is the local terminology created either by the individual translator or by groups of translators working on a common project or in the same unit. The PC-based client-server architecture of the computing infrastructure offers the possibility of using local terminology management systems in order to exploit the local reserves. However, this brings with it questions of validation and of how to encourage the individual translator to make his terminology available which are currently being attacked.

There are almost as many ways of tackling the process of translation as such as there individual translators. Most still dictate their translations, although substantial numbers now work directly onto a key-board. Whether dictating or typing, some will prepare a very rough first draft and go back over it a number of times, others will produce an almost final version straight away. Some will work right through to the end before carrying out any modifications, others will continually back-track over what has already been done. Both preferred ways of working and the urgency of a translation influence the choice between dictating and typing. Both those who preferred typing and those who preferred dictating showed a lively interest in voice dictation systems, which seemed to many to offer the best of both worlds, the speed and ease of dictation being combined with immediate availability of a text in electronic form. Whether dictating or typing, it is important to be able to concentrate and not to suffer from distractions which also come in many forms: ergonomically unsatisfactory working conditions were amongst those frequently cited.

Almost inevitably, communication and work-flow problems were also present in the life of the individual translator. Where there was a problem with the original text it was sometimes difficult to find the person in the requesting service who could resolve it, and this could have consequences on the flow of work. Translators also sometimes found it frustrating that each individual translator was trying to resolve a problem which they felt could have been dealt with centrally, either in the requesting service before the document was transmitted or higher up in the chain inside the SdT. Translators were also aware of the risk of work being done more than once, when a document contained recycled text or when version control had broken down. In this context they were keenly aware of the possibilities offered by electronic archiving, and eager for the development of good indexing and search tools.

Translators may be involved in a very wide range of activities outside the work of translation, ranging from involvement in the Permanent Delegation of Translators, which plays a highly visible role in representing staff views to the management, to acting as computing, terminology or training correspondent or helping with the evaluation or development of new tools. Although such activities are appreciated as adding interest to the working day, those who are involved in them sometimes feel that they are resented by their colleagues as not pulling their full weight in the production line, or worry that their involvement is not directly reflected in any quantified statement of their own productivity, such as the number of pages of translation produced.

The next critical link in the document production chain is the secretaries, who type any text which has been dictated and are responsible for modifications in subsequent versions or as a result of revision. The SdT experiences some difficulty in keeping secretarial staff, partly because the SdT is the most typical first post for new arrivals, partly because audio and copy typing is not very interesting work. Some units consequently have recourse to freelance typists, which can once again lead to work-flow problems, especially in those cases where the freelance typists return all but the final versions of their work on paper rather than on diskettes, so that corrections and modifications cannot be done within the Service but have to be sent back to the freelance. This problem is exacerbated by the fact that the original author is not usually easily available to the freelance typist to clear up doubts or questions.

Policy about revision varies enormously from unit to unit. There are units where every document is revised: other units are obliged to be selective. In some units, the revisor is always a more senior person, in others revision is a co-operative effort between equals based on four eyes seeing more than two. Freelance work can be a special worry, especially in those units where it is only possible to carry out spot checks. It was noticeable that revisers, perhaps even more than translators, need good tools and resources for documentation and research.

A finished translation is returned by e-mail to the requesting service. Typically, the SdT will receive no feedback on whether the job has been well done. The requester himself obviously cannot comment on versions into languages he does not know, and there is no obvious way of obtaining feedback from the end users of the translation. Many of the participants found this frustrating, and some had found ways of their own to bridge the gap, for example by occasionally attending meetings of the Committees who used the documents. Even so, there was a very common perception of a great void at both ends of the document production chain, to which the many suggestions that have been made both officially and unofficially of ways to bring requesters and translators closer together bear witness.

- Study of the document production chain closed with an account of support activities.

  Terminology is obviously of critical importance, and the heavy use of EURODICAUTOM has already been mentioned. The difficulties of maintaining and updating EURODICAUTOM were of particular concern to the terminologists, although it was pointed out that the recent implementation of batch access to EURODICAUTOM is of some help here. There was also considerable discussion of what a successor to EURODICAUTOM should look like. The issue of how best to validate and exploit local terminology naturally surfaces again in this context. The terminologists also make heavy use of other data bases, with RAPID, the data base of press releases, and various corpora of newspapers being frequently mentioned. Both terminologists and some individual units have frequent contact with specialist bodies outside the Commission, in some cases in the form of systematic collaboration on terminology acquisition and validation. Some terminology work is contracted out to freelances, essentially as piecework. This can lead to problems of quality control. Individual translators can request help with terminology through the terminology help desk, whose staff point out that experience with EURODICAUTOM teaches them "tricks" of access, for example through the subject codes, which can considerably enhance the hit rate and reduce noise.

The services of specialist documentalists are also available, and we noticed that the role of documentation specialists grows ever greater with the explosion of the information society. The documentalists make very heavy use of data bases, both internal to the Commission and on the exterior. One problem with access to data bases is that each has its own interface and mode of access: the aid of specialists here was much appreciated. For acquisition purposes, the documentalists are in permanent contact both with the requesting services and with national bodies. The clientèle for specialist documentation help is very wide: translators and terminologists, officials outside the SdT, interpretors and freelances all call on these services. This can lead to problems of knowing what requests should be fulfilled, and even of knowing who has a right to what documentation.

Training is a constant requirement within the SdT, but we noticed that the migration to a new computing architecture had required a special effort. Also, the introduction of any new tool brings with it training needs, even if the tool is only being temporarily installed for evaluation purposes. In addition to special efforts triggered by change, regular courses are offered on the use of CELEX, on project management and on the new tools already in service and their use.

Help and advice on computing facilities and on the use of tools is available through the computing correspondents and the computing help desk. There are both full-time computing correspondents who are also help desk staff and part time computing correspondents in the departments and in the individual units.

Staff from both the horizontal support units and from the translation units co-operate in the evaluation of new tools, amongst which translators workbenches and local terminology management systems are currently receiving attention. A number of issues are related to the evaluation and integration of new tools. First, there is some danger of unrealistic expectations from the side of the users. The unwary might think that a local terminology management system actually contained some terminology, rather than being a black box into which the terminology must be put, or that a translation memory came equipped with a set of example translations. Secondly, the market as a whole is geared much more to the individual translator or to a small group of translators than to the special needs of a large translation service. Thus, issues like validation of terminology or of the translations to be stored in a translation memory are largely neglected by the products on the market. Furthermore, integration of a new product into an already existing large computing service causes special problems. The individual manufacturer is, at best, concerned with compatibility only across his own range of products. A working translator cannot be expected to familiarize himself with a whole variety of different products, each of which has its own interface, its own set of commands and its own peculiarities. The EURAMIS interface created within SdT and currently on experimental release attempts to palliate these problems by providing a single uniform interface. A project of the same name jointly directed by D.G. XIII and the SdT aims at giving access to a wider range of tools and resources through a single uniform interface.

The evaluation of new tools will now be further pursued in the translation workshop (Atelier de Traduction) in Brussels and the Modernization Network (Réseau de Modernisation) in Luxembourg. These newly created organisational structures will put the new tools to use in a productive work environment.

The SdT has its own computing service. Many of the concerns of this service have already been touched upon. It is however worth adding here that market forces also have an impact on the effort to keep all languages equal. It is, for example, easy to find a text-processor with an English or French interface, but much more difficult to find one with a Greek or Dutch interface.

The size and complexity of the computing infrastructure means that the SdT has to adopt a somewhat conservative policy towards updating the services offered. Quite apart from the capital cost, the translator using the services cannot be expected to learn a new system every few months, and his productivity would suffer should he attempt to do so. Relative conservatism combined with the new PC-based client-server architecture is responsible for some tensions between the end users and the computing service. Even though the vast majority of the potential end user population have no ambition to become computing experts, a lively and articulate subset are increasingly aware of the possibilities offered by a PC. The computing services must struggle to find a compromise between giving the knowledgeable what they want, giving the majority what they need and are prepared to use, maintaining compatibility of both computing resources and linguistic resources and being able to ensure maintenance of what is offered: a delicate and challenging task.

- The final section of the report gives a list of dream tools. These are the product of asking the people interviewed to give free reign to their imagination, and suggest anything whatever, realistic or not, that would help to improve their ability to do their job. Some of the desires can already be fulfilled by the use of existing tools, or by relatively minor modifications of tools which already exist. Others, whilst not freely available, present no real technological challenge. Yet others would require major breakthroughs in fundamental research. Imposing such a classification on the list would imply considerable reflection and perhaps even research. Therefore, the tools are simply listed.

K.7 Closing remarks

I am acutely aware of the limitations of the study reported here, but I might perhaps be allowed to record some personal reflections stimulated by it.

I want first to record how fascinating I found it to do this work: as the interviews progressed I became more and more enthusiastic about the job, and realised increasingly just how complex the SdT is and how different from any other translation service I have come across.

I should perhaps apologise for adding that I was agreeably surprised by the extent to which computer tools were available and were used. Even amongst those participants who did not make heavy use of computer aids, there seemed to be a strong awareness of their utility. I particularly enjoyed those participants who started by telling me that they were radically opposed to modernization and then went on to reveal an understanding and use of the tools available considerably above average, and those who told me proudly that they never touched a computer and went on to tell me how invaluable the help of their colleagues who did use them was.

My surprise perhaps came partly from the striking difference between the SdT of today and the SdT when I was last in close contact with it. In the early 1980's I was invited by the then Head of Translation Services in Luxembourg to give a series of seminars for translators and terminologists on machine translation and machine assisted translation. (Terminology Bulletin No. 40, 1981). The audience, on the whole, was interested and keen to understand the state of the art, but very few had any direct experience of using a computer for any purpose whatever, many were rather frightened at the idea of even touching a keyboard, and a few were downright hostile to the whole enterprise. Fifteen years is not very long in the lifetime of a large institution: the attitudes then and today are light years away from each other.

I hope that these few remarks will not be perceived as impertinence: I was immensely impressed by the SdT and by its ability to tackle a gargantuan task, and am grateful to have been allowed to observe at least something of how it is done.

Appendix L

EAGLES Comments Cover Form

```
EAGLES COMMENTS --- COVER FORM

Please fill out Parts A, B and C if you are MAKING COMMENTS.
Please attach whole form at head of comments.

A. COMMENTS MADE BY:

Name:                                      Date: (0-31)/(0-12)/199_
Affiliation:
E-mail address:
Telephone:
Fax:
Postal address:


C. COMMENTS FOLLOW ON:
                                                    (delete as appropriate)

1. Text Corpora WG                                          YES     NO
   (SEND COMMENTS TO: tcwgeditor@tnos.ilc.pi.cnr.it)

2. Computational Lexicons WG                                YES     NO
   (SEND COMMENTS TO: clwgeditor@tnos.ilc.pi.cnr.it)

3. Computational Linguistics Formalisms WG                  YES     NO
   (SEND COMMENTS TO: clfwgeditor@tnos.ilc.pi.cnr.it)

4. Evaluation of NLP Systems WG                             YES     NO
   (SEND COMMENTS TO: ewgeditor@tnos.ilc.pi.cnr.it)
5. Spoken Language Systems WG                               YES     NO
   (SEND COMMENTS TO: slwgeditor@tnos.ilc.pi.cnr.it)

6. General aspects of EAGLES                                YES     NO
   (SEND COMMENTS TO: eagles@icnucevm.cnuce.cnr.it)

THANK YOU FOR YOUR TIME AND VALUABLE CONTRIBUTION TO EAGLES
```

```
D. EAGLES ACTIONS ON COMMENTS

1. Acknowledgement sent on:          by:

2. Action on comments:

   a. Dealt with by central editors directly on:

   b. Dealt with by internal editor directly on:

   c. Distributed for discussion by core WG on:

      Core WG decision:

   d. Sent to responsible of subgroup:               on:

      Subgroup decision:


= COMMENTS FOLLOW = COMMENTS FOLLOW = COMMENTS FOLLOW = COMMENTS FOLLOW =
```

References

Bibliography

Ackerman, A., Fowler, P. and Ebenau, R. (1984). Software inspection and the industrial production of software, *Software Validation. Proc. Symp. Software Validation* pp. 13–40.

Ahmad, K. (1993). Terminology and knowledge acquisition: A text-based approach, *Proceedings of Terminology and Knowledge Engineering.*

Ahmad, K., Holmes-Higgin, P., Rogers, M., Höge, M., Le-Hong, K., Huwig, C., Kese, R. and Mayer, R. (1993). User-driven software development: Translator's workbench - an exemplar case study., *in* M. Smith and G. Salvendy (eds), *Proceedings of the fifth International Conference on Human-Computer Interaction, (HCI International '93), Orlando, Florida, August 8 - 13*, Vol. 1, pp. 319–324.

ALPAC (1966). Languages and machines: Computers in translation and linguistics, *National Research Council Publication 1416*, National Research Council, Washington, DC. Report of the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences.

Athappily, K. and Galbreath, R. (1986). Practical methodology simplifies DSS software evaluation process, *Data Management* **24**(2): 10–28.

Balkan, L., Netter, K., Arnold, D. and Meijer, S. (1994). TSNLP - test suites for natural language processing, *Proceedings of the Language Engineering Convention*, ELSNET, Paris, pp. 17–22.

Beerepoot-Sangen, Y. and Leentvaar-Leistra, G. (1991). *Consument en produktkwaliteit*, Kluwer, Deventer.

BNC (1995). British National Corpus. On-line information available at.
*http://info.ox.ac.uk/bnc/

Boehm, B. W. (1988). A spiral model of software development and enhancement, *IEEE Computer* **21**(2): 169–208.

Boisen, S. and Bates, M. (1992). A practical methodology for the evaluation of spoken language systems, *Proceedings of the Third Conference on Applied Natural Language Processing, Trento*, pp. 162–169.

Boose, J. and Gaines, B. (1990). Knowledge acquisition for knowledge-based systems, *AI Newsletter* **3**.

Box, J. (1979). Konsument en informatie - de rol van vergelijkend warenonderzoek, *Thesis*, Delftse Universitaire Pers., Delft.

Bukowski, J. (1987). Evaluating software test results: A new approach, *Proceedings Annual Reliability and Maintainability Symposium, Philadelphia, USA, 27 -29. Jan*, pp. 369–375.

Capon, N. and Burke, M. (1980). Individual, product class and task related factors in consumer information processing, *Journal of Consumer Research* **7**(Dec): 314–326.

Card, S., Moran, T. and Newell, A. (1983). *The psychology of human-computer interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Carpenter, B. (1992). *The logic of typed feature structures*, Cambridge tracts in theoretical computer science, Cambridge University Press, Cambridge.

Cary, R. and Sproles, G. (1978). Evaluating product testing methods: A theoretical framework, *Home Economics Research Journal* **7**: 66–75.

Chandler, R. (1989). Grammar problems?, *Electric Word* **Sept-Oct 1989**.

Chinchor, N. (1991). MUC-3 evaluations metrics, *Proceedings of the Third Message Understanding Conference (MUC-3)*, Morgan Kaufmann, San Mateo, CA, pp. 17–24.

Cohen, G. (1980). Reading and searching for spelling errors, *Cognitive Processing in Spelling* .

Cordingley, E. S. (1989). Knowledge acquisition techniques for knowledge-based systems, *in* D. Diaper (ed.), *Knowlege elicitation principles, techniques and applications*, Ellis Horwood Limited, Chichester, UK, pp. 89–173.

Crellin, J., Horn, T. and Preece, J. (1990). Evaluating evaluation: A case study of the use of novel and conventional evaluation techniques in a small company, *in* D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), *Human Computer Interaction - INTERACT '90*, Elsevier, Amsterdam, pp. 329–335.

Cude, B. (1980). An objective method of determining the relevancy of product characteristics, *ACCI-proceedings 1980*, pp. 111–116.

Cuthbert, J. (1979). Testing for consumers, *Proceedings of the First North American Conference of Consumer Product Testing*, Ottawa Consumers' Association of Canada, Ottawa, pp. 9–21.

Damerau, F. (1980). The transformational question answering system: Description, operating experience and implications, *Report RC8287*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

Deutsch, M. (1982). *Software Verification and Validation*, Englewood Cliffs, NJ 07632.

Douglas, S. (1990). Intelligent text processing: A survey of the available products, *Hcrc technical report*, University of Edinburgh, Edinburgh.

EAGLES (1994). Evaluation of natural language processing systems, *EAG-EWG-PR.2*, ILC–CNR, Pisa.

Ericson, K. A. and Simon, H. A. (1984). *Protocol Analysis: verbal reports as data*, MIT Press, Boston.

Færch, C., Haastrup, K. and Phillipson, R. (1984). *Learner Language and Language Learning*, Nordisk Forlag, Copenhagen.

Fagan, M. (1976). Design and code inspection to reduce errors in program development, *IBM System Journal* **15**(3).

Falkedahl, K. (1991). Evaluation methods for machine translation systems: An historical overview and a critical account, *Report to Suissetra*, ISSCO, Geneva.

Falkedal, K. (1994a). Evaluation methods for machine translation systems: An historical overview and critical account, *Issco draft report*, University of Geneva, Geneva.

Falkedal, K. (ed.) (1994b). *Proceedings of the evaluators' forum, Les Rasses*, ISSCO, University of Geneva, Geneva.

Flickinger, D., Nerbonne, J., Sag, I. and Wasow, T. (1987). *Toward evaluation of NLP systems*, Hewlett Packard Laboratories, Palo Alto, CA.

Flower, L. and Hayes, J. (1980). The dynamics of composing, making plans and juggling constraints, *in* L. Gregg and E. Steinberg (eds), *Cognitive processes in writing*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 31–50.

Fulford, H. and Höge, M. (1989). Preliminary study of user requirements - methods of investigation, *Internal report of the ESPRIT II project 2315 translator's workbench (TWB)*, University of Surrey, Stuttgart and Guildford.

Fulford, H., Höge, M. and Ahmad, K. (1990). User requirements study, *Final report of the ESPRIT II project 2315 translator's workbench (TWB)*, EC, Stuttgart and Guildford.

Gaines and Shaw (1993). Basing knowledge acquisition tools in personal construct psychology, *Knowledge Engineering Review* **8(1)**: 49–85.

Galliers, J. and Jones, K. S. (1993). Evaluating natural language processing systems, *Technical report no. 291*, University of Cambridge Computer Laboratory, Cambridge.

Geistfield, L. V., Sproles, G. B. and Badenhop, S. B. (1977). The concept and measurement of a hierarchy of product characteristics, *Advances in Consumer Research* **IV**: 302–307.

Gotel, O. and Finkelstein, A. (1993). An analysis of the requirements traceability problem, *Proceedings of the International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs.

Graesser, A. (1978). Fish: The memory and representation of common procedures, *Discourse Processes* **1**: 79–89.

Grainger, S. (1993). International corpus of learner english, *in* J. Aarts, P. de Haan and N. Oostdijk (eds), *English Language Corpora: Design, Analysis and Exploitation. Papers from the thirteenth International Conference on English Language Research on Computerized Corpora*, Rodopi, Amsterdam.

Groenenveld, J. (1984). *Simple tests manual*, Consumentenbond/IOCU, 's-Gravenhage.

Gruber, T. (1989). *The acquisition of strategic knowledge*, Academic Press, San Diego.

Harman, D. (in press). The first text retrieval conference (trec1), *National Institute of Standards and Technology special publication 500-207*, NIST, Gaithersberg, MD.

Hausen, H. (1984). Comments on practical constraints of software validation techniques, *Proceedings of symposium on software validation.*, pp. 323–333.

Hausen, H. and Müllerburg, M. (1982). Kombination von verfahren fur die software-prufung, *Internationaler Kongress fur Datenverarbeitung und Informationstechnologie (IKD)* pp. 111–125.

Hausen, H., Müllerburg, M. and Schmidt, M. (1987). Uber das prufen, messen und bewerten von software. methoden und techniken der analytischen software-qualitatssicherung, *Informatik Spektrum* **10**(3): 123–144.

Hayward, S., Breuker, J. A. and Wielinga, B. J. (1987). The KADS methodology: Analysis and design for knowledge based systems, *ESPRIT P1098 Deliverable Y1*, STC Technology Ltd., Alborg.

Hellman-Tuitert, G. and Kanis, H. (1983). Produktonderzoek ten behoeve van produktinformatie, *SWOKA-onderzoeksrapport* **18**.

Hofmann, U. and Heino, H. (1992). Maschinelles übersetzen — vorteile und grenzen, *TEKOM Nachrichten der Gesellschaft für technische Kommunikation* .

Höge, M., Hohmann, A. and Le-Hong, K. (1993). User-centered software development and evaluation, *Poster Sessions. Abridged Proceedings of the fifth International Conference on Human-Computer Interaction, (HCI International '93), August 8 - 13, 1993*, Orlando, Florida, p. 166.

Höge, M., Hohmann, A. and Mayer, R. (1992). Evaluation of TWB - operationalization and test results, *Final report of the ESPRIT II project 2315 Translator's Workbench (TWB)*, Fraunhofer Society IAO and Mercedes-Benz AG, Stuttgart.

Höge, M., Hohmann, A., van der Horst, K., Evans, S. and Caeyers, H. (1993). User participation in the TWB II project - the first test cycle, *Report of the ESPRIT II project 6005 Translator's Workbench II (TWB II)*, Mercedes-Benz AG, SITE and CEC Language Services, Stuttgart, Paris, Luxembourg.

Höge, M. and Kroupa, E. (1991). Towards the design of a translator's workstation - organisational background and user implications, *in* H.-J. Bullinger (ed.), *Human Aspects in Computing: Design and Use of Interactive Systems and Information Management, 18B. Proceedings of the Fourth International Conference of Human-Computer Interaction, Stuttgart, Germany*, Elsevier, Amsterdam, pp. 1036–1040.

Höge, M., Wiedenmann, O. and Kroupa, E. (1991). Evaluation of the TWB — theoretical framework and practical application, *Report of the ESPRIT II project 2315 translator's workbench (TWB)*, EC, Stuttgart.

Hohmann, A., Le-Hong, K. and van der Horst, K. (1994). User participation in the TWB II project - the second test cycle, *Report of the ESPRIT II project 6005 Translator's Workbench II (TWB II)*, Mercedes-Benz AG and CEC Language Services, Stuttgart and Luxembourg.

Hönig, H. and Kusmaul, P. (1982). *Strategie der Ubersetzung. Ein Lehr und Arbeitsbuch*, Somepublisher, Tubingen.

Howden, W. (1980). Functional program testing, *IEEE Transactions on Software Engineering* **6**: 162–169.

Hutchins, W. (1986). *Machine translation*, Ellis Horwood, Chichester.

Hüttenrauch, R. (1977). Zur methodik des vergleichenden warentests, *Journal of Consumer Policy* **2**: 143–150.

IOCU (1977). *Comparative Testing Guide*, IOCU Testing Committee IOCU, The Hague.

IOCU (1985). *Guide to the Principles of Comparative Testing*, IOCU Testing Committee IOCU, Penang.

ISO (1991). *International Standard ISO/IEC 9126. Information technology — Software product evaluation — Quality characteristics and guidelines for their use*, International Organization for Standardization, International Electrotechnical Commission, Geneva.

Jackel, B. (1989). Methods of market research for comparative testing, *Report of the Seminar on 'Market Research for Comparative Product Testing'* .

Jackson, M. (1995). Problems and requirements, *Proceedings of the Second IEEE International Symposium on Requirements Engineering, York, England*, IEEE Computer Society Press, Los Alamitos, California, pp. 2–9.

Jarke, M., Turner, J., Stohr, E., Vassiliou, Y., White, N. and Michielsen, K. (1985). A field evaluation of natural language for data retrieval, *IEEE Transactions on Software Engineering* **SE-11**(1): 97–113.

JEIDA (1992). *JEIDA methodology and criteria on machine translation evaluation*, JEIDA, Tokyo.

Kanis, H. (1988). De consumentenbeweging als professionele klant, *SWOKA-onderzoeksrapport* **49**.

Karat, C. (1990). Cost-benefit analysis of iterative usability testing, *in* D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), *Human Computer Interaction - INTERACT '90*, Elsevier, IFIP, pp. 351–356.

Kelly, G. A. (1955). *The Psychology of Personal Constructs*, Norton, New York.

King, M. and Falkedal, K. (1990). Using test suites in the evaluation of machine translation systems, *Proceedings of COLING-90*, ACL, Helsinki, pp. 211–219.

LDC (1995). Canadian Hansard – Parliamentary Proceedings. On-line information available at. *http://www.ldc.upenn.edu/ldc/catalog/html/listing.html

Le-Hong, K., Höge, M. and Hohmann, A. (1992). User's point of view of the translator's workbench, *Translating and the Computer. Quality Standards and the Implementation of Technology in Translation. ASLIB, 10-11 November 1992* **14**: 25–31.

Lehnert, W. and Sundheim, B. (1991). A performance analysis of text-analysis technologies, *AI Magazine* **12**(4): 81–94.

Lewis, J., Henry, S. and Mack, R. (1990). Integrated office software benchmarks: A case study, *in* D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), *Human Computer Interaction - INTERACT '90*, Elsevier, Amsterdam, pp. 337–343.

Liberman, M. (1989). Text on tap: the ACL/DCI, *Proceedings of the DARPA Speech and Natural Language Workshop*, Morgan Kaufmann, San Mateo, CA, pp. 173–188.

Meyer, I., Bowker, L. and Eck, K. (1992). COGNITERM: An experiment in building a terminological knowledge base, *Proceedings of the 5th Euralex International Congress.*

Mikheev, A. and Moens, M. (1994). KADS methodology for knowledge-based language processing systems, *in* B. R. Gaines and M. Musen (eds), *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 5–1–5–17.

Miller, E. (1984). Quality managment technology: Practical applications, *Software Validation* pp. 255–266.

Miller, E. and Howden, W. (eds) (1981). *Intorial: Software Testing and Validation Techniques*, IEEE, London.

Moll, T. and Ulich, E. (1988). Einige methodische fragen in der ananlyse von mensch-computer interaktion, *Zeitschrift fur Arbeitswissenschaft* **42**(2): 70–76.

MT (1994). on evaluation, *Machine Translation* **8**(1.2). Edited by D. Arnold, R.L. Humphreys and L. Sadler.

MUC-3 (1991). *Proceedings of the Third Message Understanding Conference (MUC-3)*, Morgan Kaufmann, San Mateo, CA.

Murine, G. and Carpenter, C. (1983). Applying software quality metrics, *Proceedings from the ASQC Quality Congress, Transactions*, Boston.

Musa, J., Iannino, A. and Okumoto, K. (1987). *Software Reliability, Measurement, Prediction, Application*, Somepublisher, New York.

Myers, J. H. and Shocker, A. D. (1981). The nature of product-related attributes, *Research in Marketing* **5**: 211–236.

Nagao, M., Tsuji, J. and Nakamura, J. (1988). The japanese government project, *in* J. Slocum (ed.), *Machine translation systems*, CUP, Cambridge.

Neal, A. and Simons, R. (1985). Playback: A method for evaluating the usability of software and its documentation, *Proceedings of the Anniversary Meeting 1985, User Friendly Computing September 23-27, 1985*, Vol. 2, pp. 1051–1075.

Norman, D. (1985). Four stages of user's activities, *Proceedings of Human-Computer Interaction — Interact'84*.

Nuseibeh, B., Kramer, J. and Finkelstein, A. (1994). A framework for expressing the relationships between multiple views in requirements specification, *IEEE Transactions on Software Engineering* **20**(10): 760–773.

O'Connell, T., O'Mara, F. and White, J. (1994). The ARPA MT evaluation methodologies: Evolution, lessons and further approaches, *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, Columbia, U.S.A.

Oppermann, R., Murcher, B., Pateau, M., Pieper, M., Simm, H. and Stellmacher, J. (1988). *Evaluation von Dialogsystemen. Der software-ergonomische Leitfaden EVADIS*, Somepublisher, Berlin.

Osterweil, L. (1984). Integrating the testing, analysis and debugging of programs, *in* H. Hausen (ed.), *Software Validation*, Amsterdam, North-Holland, pp. 73–93.

Rahmstorf, G. (1994). RELATIO/IR, *Workshop on semantic information retrieval*, IBM, Heidelberg.

Rahmstorf, G. and Rabinovitz, R. (1993). Better writing through electricity, *PC Magazine* **May 1993**: 147–200.

Roman, G. (1985). A taxonomy of current issues in requirements engineering, *IEEE Computer* **April**: 14–22.

Ross, E. (1991). Zielgruppenanalyse für die technische dokumentation mit hilfe des stereotypenansatzes als methode der künstlischen intelligenz, *Proceedings of GAL-91, Mainz, Germany.*

Ross, E. (1993). IASTA: An intelligent assistant system for technical authors, *Proceedings of the 4th Annual Conference on Online Documentation, München, Germany.*

Ross, E. (in press). *IASTA – An intelligent assistant system for technical authors*, Ph.D. dissertation, University of the Saarland, Saarbrücken. Forthcoming.

Ross, E. and Verdret, P. (1992). From text to hypertext: Methodology and techniques for converting software documentation, *Proceedings of the 3rd Annual Conference on Online Documentation, Böblingen, Germany.*

Rushinek, A. and Rushinek, S. (1985). Accounting and auditing software evaluation with knowledge based expert systems: An empirical multivariate model, *Fourth Annual International Conference on Computers and Communications '85, Conference Proceedings, March, 20-22, 1985*, pp. 250–254.

Russo, J. E. (1987). Toward intelligent product information systems for consumers, *Journal of Consumer Policy* **10**: 109–138.

Russo, J. E. (1988). Information processing from the consumer's perspective, *Proceedings of the International Conference on Research in the Consumer Interest*, pp. 185–217.

Salton, G., Buckley, C. and Smith, A. (1990). On the application of syntactic methodologies in automatic text analysis, *Information Processing and Management* **26**(1): 73–92.

Schmied, W.-S. and Winkler, H. (1989). *Software-Qualität. Ausgewählte Methoden und Werkzeuge der Softwareprüfung*, Siemens-Schriftenreihe data praxis, Siemens, München.

Schütz, J. and Ripplinger, B. (1993). Machine translation supported by terminological information, *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation, Kyoto, Japan*, pp. 102–116.

Shaw, M. L. G. and Gaines, B. R. (1989). A methodology for recognising conflict, correspondence, consensus and contrast in a knowledge acquisition system, *Knowledge Acquisition* **1**(4): 341–363.

Sherman, B. and Judkins, P. (1993). *Glimpses of heaven, visions of hell — Virtual reality and its implications*, Coronet Books, Hodder and Stoughton Ltd, London.

Siderius, P. (1989). Produktinformatiesystemen, *SWOKA-onderzoeksrapport* **82**.

Silberer, G. (1985). The impact of comparative product testing upon consumers. selected findings of a research project, *Journal of Consumer Policy* **8**: 1–27.

Silberer, G. and Raffé, H. (1984). *Warentest und Konsument, Nutzung, Wirkungen und beurteilung des vergleichenden Warentests im Konsumentenbereich*, Campus Verlag, Frankfurt.

Skuce, D. and Lethbridge, T. (1993). Code4: A multifunctional knowledge management system, *Department of computer science*, University of Ottawa, Ottawa.

Slage, J. and Wick, M. (1988). A method for evaluating expert system applications, *AI Magazine* **9**.

Sneed, H. (1987). Software-testen - state of the art, *Software Entwicklungs-Systeme und Werkzeuge, 2 Kolloquium, 8-10, September 1987* .

Steenkamp, J. B. E. M. (1989). *Product quality: an investigation into the concept and how it is perceived by consumers*, van Gorcum, Assen/Maastricht.

Sundheim, B. (1991). Overview of the third message understanding evaluation and conference, *Proceedings of the Third Message Understanding Conference (MUC-3)*, Morgan Kaufmann, San Mateo, CA, pp. 3–24.

Tansley, D. S. W. and Hayball, C. C. (1993). *Knowledge Based Systems Analysis and Design: A KADS Developer's Handbook*, Prentice Hall, Englewood Cliffs, NJ.

Thaller, G. (1993). *Qualitätsoptimierung der Software-Entwicklung. Das Capability Maturity Model (CMM)*, Somepublisher, Braunschweig/Wiesbaden.

Thaller, G. (1994). *Verifikation und Validation. Software Tests für Studenten und Praktiker*, Somepublisher, Wiesbaden.

Thompson, H. (1992). The strategic role of evaluation in natural language processing and speech technology, *Record of the ESPRIT/DANDI/ELSNET/HCRC Workshop, Edinburgh 1992*, Human Communication Research Centre, University of Edinburgh, Edinburgh.

Thompson, H. (1994). TEMAA : A testbed study of evaluation methodologies : Authoring aids, *Proceedings of the Language Engineering Convention*, ELSNET, Paris, pp. 147–148.

Thorelli, H. B. (1979). The future for consumer information systems, *in* W. L. Wilkie (ed.), *Advances in Consumer Research*, Vol. 6, Association for Consumer Research, Ann Arbor, pp. 227–232.

Thorelli, H. B. and Engledow, J. L. (1980). Informationseekers and information systems: a policy perspective, *Journal of Marketing* **44**(Spring): 9–27.

TLF (1995). Trésor de la Langue Française. On-line information available at. *http://humanities.uchicago.edu/ARTFL/ARTFL.html

Vainio-Larsson, A. (1990). Evaluating the usability of user interfaces: Research in practice, *in* D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), *Human Computer Interaction - INTERACT '90*, Elsevier, Amsterdam, pp. 323–328.

van Weperen, W. (1985). Certification and product testing. some problems in consumer information systems, *Journal of Consumer Policy* **8**: 373–388.

Whittaker, S. and Walker, M. (1989). Comparing two user-oriented database query languages: A field study, *Technical report HPL-ISC-89-060*, Hewlett Packard Laboratories, Bristol.

Willenborg, G. (1985). Consument en produktinformatie, *SWOKA-onderzoeksrapport* **29**.

Woods, W. (1973). Progress in NLU — an application to lunar geology, *AFIPS Conference Proceedings 42*, pp. 441–450.

XeroxParc (1992). The information theater, *Internal presentation*, Xerox PARC, Palo Alto, CA.