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

All trade marks are hereby acknowledged.

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
NL-3512 JK Utrecht
The Netherlands
Tel: +31.30.536.049
Fax: +31.30.536.000
E-mail: destombe@let.ruu.nl

BECH, Annelise (Editor, preliminary report)
Centre for Language Technology
Njalsgade 80
DK-2300 Copenhagen
Denmark
Tel: +45.35.32.90.76
Fax: +45.35.32.90.89
E-mail: cstadm@cst.ku.dk

NEVILLE, Anne (Editor, final report)
Centre for Language Technology
Njalsgade 80
DK-2300 Copenhagen
Denmark
Tel: +45.35.32.90.76
Fax: +45.35.32.90.89
E-mail: anne@cst.ku.dk

ARPPE, Antti (Translators' Aids)
Institute of Industrial Management
Helsinki University of Technology (HUT)
Office U503
Otakaari 1 M
FIN-02150 Espoo
Finland
Tel: +358.0451.3088
Fax: +358.0451.3095
E-mail: Antti.Arppe@hut.fi

BALKAN, Lorna (Writers' Aids)
Department of Language and Linguistics
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
UK
Tel: +44.206.872091
Fax: +44.206.872085
E-mail: balka@sx.ac.uk

BRACE, Colin (Translators' Aids)
Language Industry Monitor
Eerste Helmersstraat 183
NL-1054 DT Amsterdam
The Netherlands
Tel: +31.20.685.0462
Fax: +31.20.685.4300
E-mail: colinb@paramount.nikhefk.nikhef.nl

BUNT, Harry (Translators' Aids)
Institute for Language Technology
and Artificial Intelligence
University of Tilburg
PO Box 90153
NL-5000 LE Tilburg
The Netherlands
Tel: +31.13.66.3060
Fax: +31.13.66.2537
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: $O \times U \rightarrow V$ 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

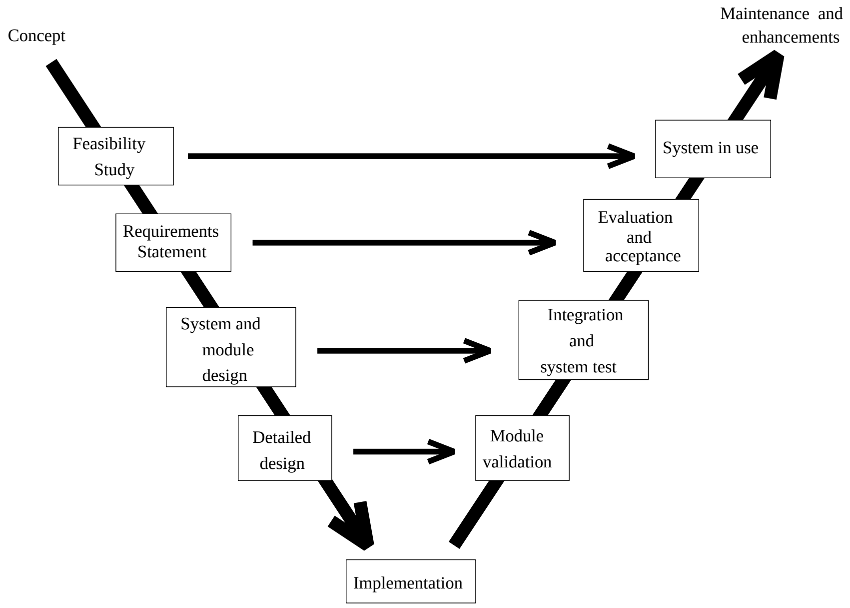
The input to the ISO process model of evaluation (2.1.1) is a statement of *stated or implied needs*. This section outlines what this should involve for language engineering products. Further detail of the approach is provided in Appendix C.

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

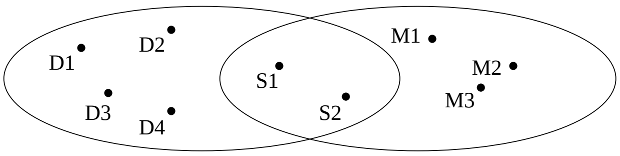


Figure 2.2: Domain and machine attributes and specifications.

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 .

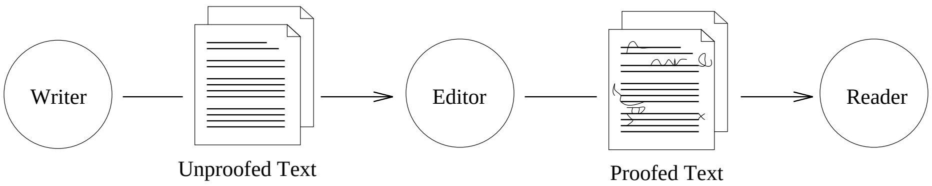


Figure 2.3: Top level data flows and agents in spell checking task.

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.

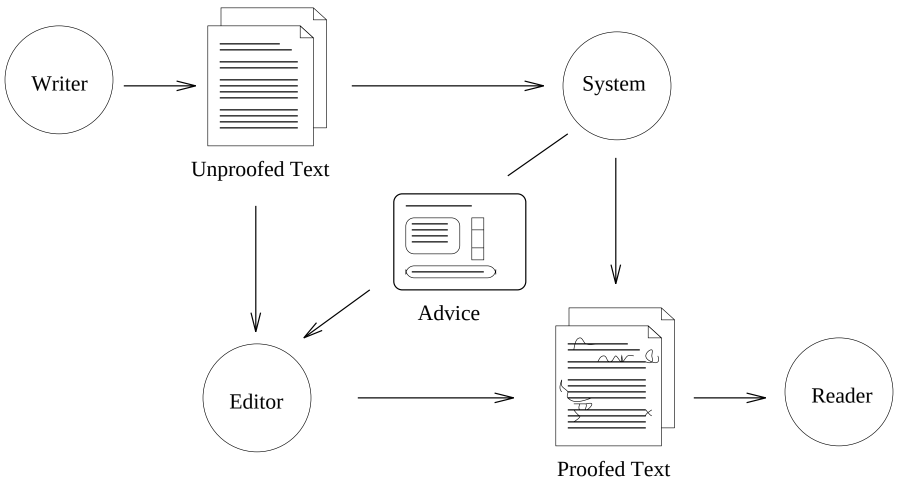


Figure 2.4: Introduction of computational system.

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 every-day 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 sub-characteristics 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.

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.