Towards a Framework for the Evaluation of Translators' Aids' Systems

Monika Höge

Department of Translation Studies

University of Helsinki

Finland

Helsinki University Press
Helsinki

ISBN 952-10-0555-6 (Print)
ISBN 952-10-0556-4 (Electronic Version)

Towards a Framework for the Evaluation of Translators' Aids' Systems

The framework for the evaluation of translators' aids makes use of an interdisciplinary approach that integrates findings from translation theory, software engineering, and decision analysis. It shows that if the different mechanisms offered by these disciplines are applied to the problem of evaluating translators' aids' systems, a better understanding of the processes that are required to arrive at reliable and valid results is achieved. To prove the feasibility of the framework empirical tests involving translators and translators' aids' systems are presented.

Evaluation is perceived as a cycle that covers the phases of featurisation, modelling, testing and assessment.

Central to featurisation is the description of the translation context. The framework provides parameters along which the features of the institutional and individual dimensions of the translation context can be elicited. They are based on considerations from both translation theory and the professional context of translation, as well as principles of requirements elicitation stemming from software engineering.

Modelling in evaluation is concerned with the structuring, categorisation and generalisation of information in order to reduce the evaluation effort. The domain model covers a description of typical tasks. The quality model covers a description of the relevant system attributes and their metrics, that is, ways of measuring attributes. The test model denotes which attributes can be measured by which type of test.

Based on software engineering principles and empirical test descriptions, three test types are described which guarantee that the user point-of-view is adequately considered: *Scenario testing* is performed to assess the appropriateness of a piece of software for every-day work; *systematic testing* is performed to examine the behaviour of software under specific test conditions; and *feature inspection* is a means to check the actual functionality of a piece of software.

The assessment procedure is based on multiattribute measurement principles discussed in decision analysis. In short, value functions are constructed for each attribute under testing, and the additive weighted model is applied to the level of translation tasks, thus providing a numeric suitability result between 0 and 100 for each translators' aids' system under evaluation.

Preface

The idea of developing a framework for evaluating translators' aids systems occurred during the ESPRIT projects TWB I and II (1989 - 1994), when Mercedes-Benz was the user-organisation on behalf of which I had to evaluate the great variety of translation tools that were developed by different European partners. Close examination of the translation process that was to be supported by the computer showed that the phenomena involved in evaluating translators' aids systems could not be adequately dealt with within the translation discipline. The lack of available methodologies and the practical need to arrive at evaluation results in the TWB projects made it imparative that a new way had to be found which integrated findings from the discipline of software engineering. The requirements investigation and tests performed in the course of the two TWB projects were merely practical, though theoretically informed, evaluation processes.

As research assistant to the University of Helsinki, my original intention was to back up and support the practical work with more theoretical considerations in the disciplines of translation theory and software engineering. Starting off from the practical angle in the TWB projects, the investigation of theoretical principles was always guided by their practical applicability. By 1997, a first draft of an evaluation framework was developed, which (i) was based on experiences gained during the evaluation of the TWB projects, (ii) integrated principles and mechanisms discussed in translation theory and software engineering, and (iii) took into account various discussions within the EAGLES evaluation group. The framework included theoretically based definitions of ways of modelling and testing requirements. The evaluation framework was mainly geared to evaluation in the software development context and was largely based on qualitative principles, describing the adequacy of a system to support a specific translation task.

At this point, the problem of assessment and the mechanisms involved in quantitatively describing degrees of quality attracted further interest. It had become obvious that neither translation theory nor software engineering could provide mechanisms to quantitatively assess the suitability of a system for a specific environment. Considerable assistance could be derived from the discipline of decision analysis which has been concerned with the evaluation of projects and educational programmes since the 1960ties. The integration of mechanisms used in decision analysis into the existing evaluation framework was more than a year's work, leading to the theoretically based and practically applied framework for the evaluation of translators' aids described in this thesis.

The thesis covers more than ten years of my professional life of which I spent half as researcher at the translation department of Mercedes-Benz and half as research assistant and postgraduate student at the University of Helsinki, performing course work or participating in EAGLES projects. I am grateful to Mercedes-Benz who allowed me to submerge myself in theory more than was normal practice for EU projects. I would also like to thank my supervisor Prof. Lauri Carlson for his support during many years of work, in which I could regularly come over to Finland to discuss progress. Thanks also to Khurshid Ahmad and Krista Varantola who evaluated the thesis.

Without the help and support of IBM and TRADOS who supplied the software for testing for free and delivered excellent hotline support, testing could not have been performed. Likewise, I want to thank Edith Kroupa at Mercedes-Benz and Kees van der Horst at the European Commission for providing me with a text corpus of real texts and their translations that were used to develop the test data for the different tests. For valuable discussions and proofreading I'd like to thank my friends Hilary Charman and Paul Holmes-Higgins.

As to the last ten years of my private life, I am very grateful that my husband Robert supported this project not only with his good will but also with literature, printouts and discussion. Apart from producing some 300 pages of text, I was also busy in that period producing two girls, Elena and Jana, who have never stopped wondering what on earth their mother was doing all these years with that grey box called computer. Thank you all for your patience with your ever-busy partner and mother.

Table of Contents

List of Figures

List of Abbreviations

CASE	computer aided software engineering
CAT	computer aided translation
EAGLES	expert advisory group on language engineering standards
ES	epistemic
EU	european union
FL	foreign language
HTML	hypertext markup language
HS	heuristic
IEEE	institute of electrical and electronics engineers
ISO	international organisation for standardisation
KA	knowledge acquisition
LE	language engineering
LGP	language for general purposes
LRE	linguistic research and engineering
MT	machine translation
NLP	natural language processing
PH	person hours
RE	requirements engineering
SE	software engineering
SGML	standard general markup language
SL	source language
ST	source text
TA	task analysis
TAKD	task analysis for knowledge descriptions
TB	termbank
TDH	task description hierarchy
TL	target language
TM	translation memory
TT	target text
XML	extended markup language

Introduction and Approach

Research and development in the area of Language Engineering is of importance in an increasingly globalised world from an economic, political, and cultural viewpoint. Many financial, commercial and industrial transactions require multilingual processing of language, and the classification, processing, storage and retrieval of documents. The European Union has launched independent programmes and initiatives for the development of Language Technologies. One area of interest within this field is the evaluation of the resulting prototypes by different user groups. It has been repeatedly found that performance measurement and evaluation is a badly neglected activity and needs to be encouraged in future programmes in relevant projects.

Tools supporting the translation process belong to the rather complex, yet rapidly growing area of Natural Language Engineering. The evaluation of these systems has so far been primarily application dependent in that for each evaluation scenario, the evaluators tailored specific procedures and techniques from the set of techniques used in other evaluations. It is important to compile a list of methods used in a range of evaluation scenarios, examine these techniques and evaluate the tools used. This compilation, examination and critique may lead to a methodology – a systematically organised set of methods, tools and techniques that introduce rigour into the evaluation of translators' aids whilst allowing the evaluator to have a choice of methods, tools and techniques. Research in this area is impeded by its interdisciplinary nature and the requirement for integrating findings from different scientific areas such as software engineering, decision analysis, and translation.

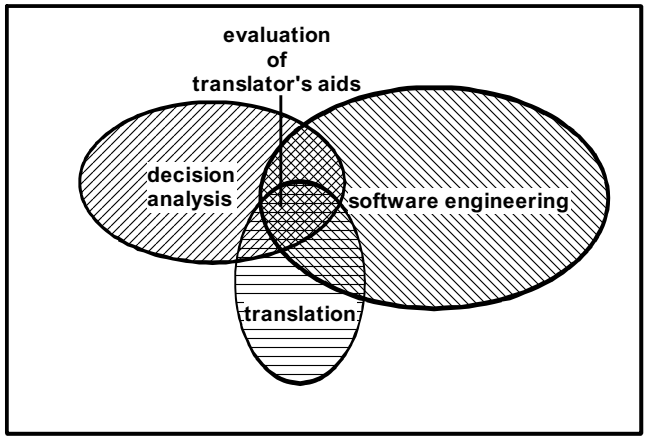


Figure 1: Disciplines Related to the Problem of Evaluation of Translators' Aids

Among the three disciplines related to the evaluation of translators' aids, *software engineering* has the largest impact on the development of a methodology for software

evaluation. Many of the procedures relevant to the development of software are, to some extent and with a different focus, also relevant to software evaluation. There are also a great many parallels between the processes involved in *decision analysis* and those involved in evaluation. Struening/Guttentag (1975) and Guttentag/Struening (1975) document experiences in which decision analysis procedures have been applied for evaluation purposes since the social science research carried out in the 1960s. Evaluators of software systems can learn from experiences made in the evaluation of social programmes, particularly with respect to the definition, organisation and weighting of attributes that are relevant when it comes to choosing between different alternatives. Most importantly, the discipline of *translation* serves as a basis for the definition of those problems that are involved in the particular instance of evaluation, that is, the evaluation of translators' aids.

Considering the above three disciplines and adapting their approaches to the problem of the evaluation of translators' aids, the approach adopted in this thesis is that of an evaluation cycle, which starts off with examining and describing features of both the user and the systems under evaluation. Followed by modelling, which involves the elaboration and structuring of the system context, the quality attributes relevant, and the test types that will allow the measurement of the required attributes. Testing delivers values for the attributes that are identified during modelling. Assessment, finally, validates test results and relates them back to the users.

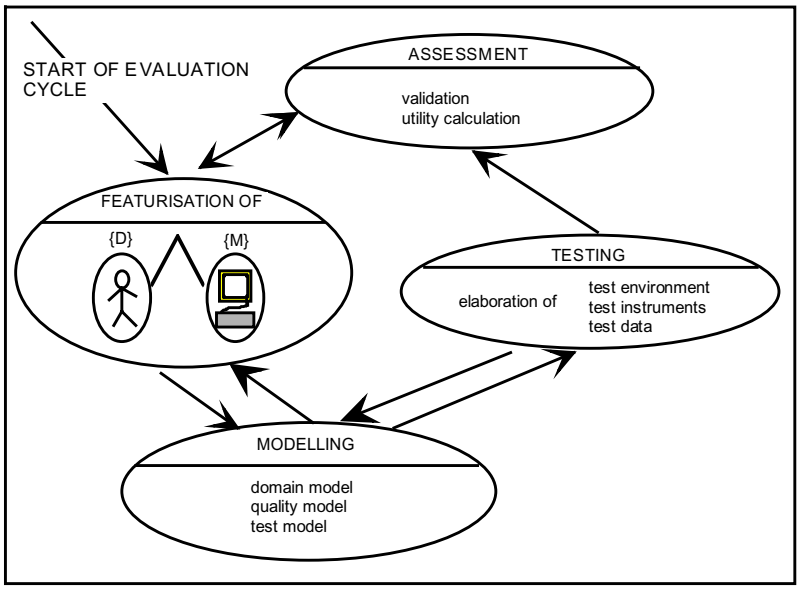


Figure 2: The Evaluation Cycle

Translation and Evaluation of Translators' Aids

The goal of the evaluation of translators' aids' systems is to help support translation work. The extent to which these systems can help translators in their everyday work can only be determined on the basis of a sound understanding of the problems involved in the translation process. Consulting relevant theoretical literature on translation and the translation process, it has become obvious that the major flaw of current investigations is that translation has mostly been considered in isolation, neglecting the fact that it is a major part of the documentation life-cycle. When it comes to defining appropriate translators' aids, the focus must not be only on the problems encountered by the translator to produce an adequate target text but also on the technical background, the strategies applied by the author of the source text, the final users of the document, and the context in which translation is carried out. In chapter 1, the nature of the translation process is described in detail with reference to both theory and practice in order to find out where and the extent to which it would make sense to support the translator with computational translators' aids.

Software Engineering and Evaluation of Translators' Aids

One of the most widely studied models of software development is the so-called life-cycle model. Based on the biological analogy of conception, birth, growth, maturity and death, leading software engineers such as Sommerville (1996[s]:9) or Thaller (1993:104) suggest that software development includes inception or refinement, specification, analysis, design, implementation, testing, delivery and operation.

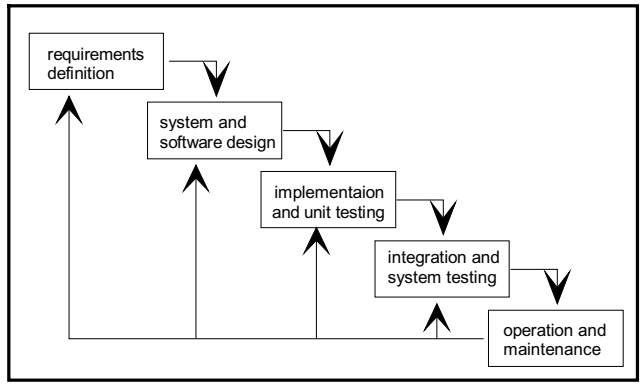


Figure 3: The 'Waterfall' Model of the Software Life Cycle by Sommerville (1996:9)

According to the IEEE (1059:9) Guide for Software Verification and Validation Plans, evaluation ascertains the value or worth of a system for a particular environment and uncovers problems in the software product which relate to the basic user need for the

system to be fit for use in its intended setting. This involves most importantly assuring that

- the product conforms to its specification
- the product is correct
- the product is complete, clear, and consistent
- the product complies with all appropriate standards
- the product meets all specified quality attributes.

In fact, many of the steps in the development cycle and the evaluation cycle have crucial problems in common. The main differences are due to the fact that in the development cycle (D) a new product is developed and in the evaluation cycle (E) a given product is evaluated. This implies differences in the order in which steps are taken, in the direction in which they are taken, and in the relative importance of the steps. The following figure describes the common features of both processes in a schema of which the two cycles are temporal projections.

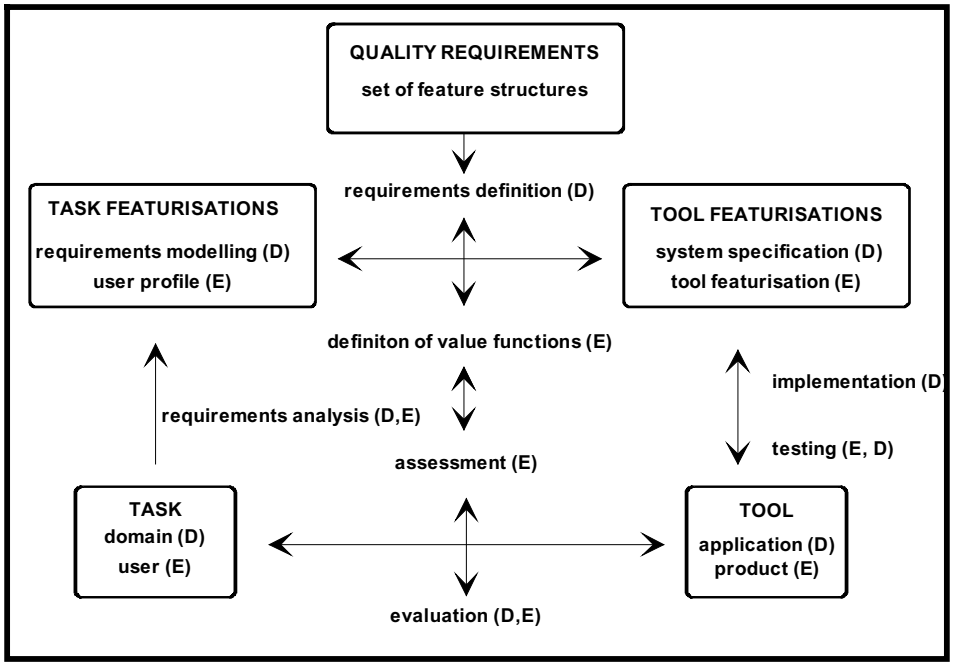


Figure 4: Combination of Development and Evaluation Cycle

In Figure 4 those aspects that are relevant to the development cycle are marked with (D) and those that are relevant to the evaluation cycle are marked with (E). The task or action, which is to be supported computationally, is a central entity in both cycles. While in the development cycle there is a focus on the description of the overall *domain* in which a task occurs, in user-oriented evaluation the *user's* view of the task is a central issue. The process of *requirements analysis* results in the featurisation of

the task, that is, a description of the features or characteristics of the task. In the development cycle the featurisation of the task is but one part of the overall process of *requirements modelling*, while in evaluation, the featurisation of the task is the central criterion for the definition of different *user profiles*, which capture regular variations in requirements. In the development cycle the next step is to define the future features of the system in terms of functional and non-functional quality requirements. This process is commonly called *requirements definition*. The definition of quality criteria in evaluation, however, asks for the mapping of tasks onto tool features, or, in other words, *value functions* are only defined for those features of a task that can be performed by a tool in one way or the other. Consequently in evaluation, the featurisation of the task has to be followed by the featurisation of the tool, that is, the description of the features or the characteristics of the tool. In the development cycle, there is also a form of tool featurisation, which results from the definition of requirements. This form of tool featurisation is called *system specification*, which forms the basis for the process of implementation. *Testing* is performed in both cycles. In the development cycle, testing is performed at the component, integration and system levels. In the evaluation cycle, the way of testing is determined by (i) the goal behind testing, that is, what do we want to achieve, and (ii) the nature of the pre-developed value functions. *Assessment* closes the evaluation cycle by measuring the extent to which the tool is capable of performing the tasks, or in other words, by mapping the results of testing onto a utility scale.

Decision Analysis and Evaluation of Translators' Aids

According to French (1986), decision analysis refers to the careful deliberation that precedes a decision, more particularly, to the quantitative aspects of that deliberation. Testing delivers values for those attributes that are identified in the modelling phase. Decision analysis may help in interpreting these values in a rational way. There are some obvious parallels between the activities involved in the evaluation cycle as described in Figure 2 and the activities involved in the decision analysis process as depicted by leading decision analysts such as Keeney/Raiffa (1993[3]); Winterfeldt/Edwards (1986).

ACTIVITIES IN DECISION ANALYSIS PROCESS	ACTIVITIES IN EVALUATION	EVALUATION PHASE
identifying the problem	definition of task and tool	featurisation phase
develop decision maker model	task and tool featurisation	
matching problems and structures	matching task and tool features	modelling phase
develop value tree	develop quality model	
define objects in value relevant terms	definition of measurable attributes	
scale construction	definition of metrics, that is a way of measuring a specific attribute, leading to an attribute/value pair	
construction of value functions	definition of target values	
utility measurement	comparison between target and actual values	assessment phase

Figure 5: Parallels between Activities in Evaluation and Decision Analysis

Winterfeldt/Edwards (1986:222) illustrate how decision analysis works by means of the following example: A new job involves house moving. Which of the apartments at hand is the best choice? The following figure shows the construction of a value function with respect to the attribute *location of apartments*.

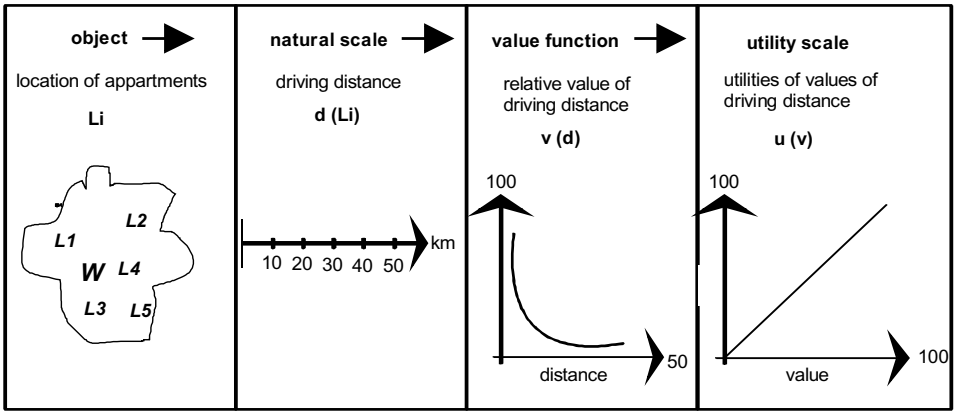


Figure 6: Construction of Value Function According to Winterfeldt/Edwards (1986:222)

Location of apartments is but one of a set of attributes that are of importance for the decision described. Other attributes in the set may be *price of apartment*, *architecture*, *environment* etc. If more than one attribute needs to be considered, the decision process roughly involves the following steps:

1. Elaborating value functions for each attribute under consideration
2. Assigning relative weights to attributes
3. Defining aggregate utility for different options
4. Relating utility to cost and performing tradeoffs

5. Performing sensitivity analysis

The contents of assessment defines how diverse values can be combined and compared. This is where the theory of measurement comes in. How can different values be compared? The principles of measurement and evaluation stemming from decision analysis will be described and employed in evaluation of translators' aids wherever possible.

To conclude, Galliers and Sparck Jones (1993:140) suggest that the principal question when elaborating a framework for user-oriented evaluation is whether evaluation criteria, measures and methods can be generalised? That is, is an evaluation necessarily task - even application - dependent, or can specific evaluation techniques, as opposed to abstract concepts, be applied across individual cases?

The difficulty in establishing an evaluation framework lies in recognising those aspects of evaluation that can be compared and used across different evaluation scenarios and in formulating guidelines on how to proceed. In this thesis it will be argued that there are three issues that may be relevant to software evaluation, specifically the evaluation of translators' aids, which are

(i) an awareness of the needs of the users of the system;
(ii) an understanding of the functionality of the system under evaluation; and
(iii) the knowledge of techniques on how to test and assess the performance of the system.

In this thesis it will be argued that the need for an evaluation framework for translators arises in two typical situations:

(i) In the translation industry, preceding a purchase decision

Translators under pressure from shortening product life-cycles and an increase in international communication do not feel they have the time to implement available systems into the existing translation environment. Purchase decisions are not infrequently postponed due to a lack of methods for evaluating different alternatives in terms of costs and benefits. Thus, despite the growing number of translation support systems offered, decision makers are reluctant to implement available systems into the existing translation environment.

(ii) In translation system development, supporting the development process

The development of translators' aids' systems requires computer-based knowledge of the practical problems faced during translation

To sum up, the outstanding objective of this work is to identify evaluation procedures, describe techniques and formulate guidelines for their use, which help evaluators and, if possible, even enable end users to perform evaluations of specific systems for their particular purposes. The framework for the evaluation of translators' aids which will be developed in this thesis

- is user-oriented, that is, geared towards user groups or user representatives as the agents of evaluation;
- focuses on aspects that may be relevant to translators as users of the tools;
- is based on findings of the three major disciplines concerned, that is, software engineering, decision analysis and, translation.

The main achievements of the framework for evaluation developed in this thesis can be summarized as follows:

i. Based on the investigation of the translation process in terms of epistemic and heuristic knowledge structures used for problem solving, a theory was elaborated why and how the translation process can be supported by the computer;
ii. Applying decision theoretic, problem-oriented analysis onto requirements engineering principles, a method was found to bridge the gap between the elaboration of user requirements and the development of measurable primitives;
iii. Taking into account basic software engineering principles, a goal-oriented model of test types for user testing was developed which bridges the gap between prototype and product testing;
iv. Applying decision analytic and measurement theoretic principles onto requirements, an assessment procedure was developed that allows to quantitatively compare the adequacy of different systems for a specific context and shows where tradeoffs have to be made in terms of costs vs. quality;
v. Procedures were developed that describe how evaluation results can be obtained by performing four major evaluation phases, that is, (i) featurisation of user and system; (ii) modelling of domain, quality and test information; (iii) testing; and (iv) assessment;
vi. Possibilities for the re-usability of resources in evaluation were identified and elaborated in terms of
    - procedures for featurisation, modelling, testing and assessment;
    - parameters relevant to the elicitation of translators' requirements;
    - metrics applicable for translators' aids' systems.

Chapter 1 provides an overview of the context of both translation and evaluation. It will be investigated the extent to which translation theory can help in defining evaluation relevant information. The nature of translation tools and the most prominent systems on the market will be described. The evaluation context is characterized by numerous international initiatives that strive to develop tools to further evaluation work. It will be pointed out what the difference is between these approaches and the framework developed for this thesis.

Chapter 2 discusses the activities that fall under the first step in the evaluation cycle, that is the elicitation and description of features of the user and the system. Parameters will be presented along which information about the domain of translators can be gathered in form of questionnaire surveys, interviews or observations.

Chapter 3 is concerned with preparing for evaluation by means of structuring the manifold information gathered during the elicitation process. It employs methods from software engineering and decision analysis, leading to a procedure for evaluation that allows the definition of measurable primitives.

In chapter 4 a test model will be presented which is based on both software engineering considerations and experiences in practical software testing. The elaboration of test data will be discussed from the angle of software and language engineering. Experiences and results of exhaustive practical testing procedures with two commercially available translators' aids' systems will be presented.

Chapter 5 concludes the evaluation cycle, showing how to adapt and make use of quantitative evaluation procedures stemming from decision analysis. The evaluation model for assessment will show that it is possible to arrive at numerical, comparable results integrating the outcome of the practical testing procedures described in chapter 4.

The thesis concludes with a summary of achievements and conclusion of the endeavour of developing a framework for the evaluation of translators' aids' systems. It will prove what Bechtel (1986:pp.30) argues, that is, crossing the disciplinary boundaries often provides a better understanding of complex problems. In other words, applying the different mechanisms offered by translation theory, software engineering and decision analysis leads to a better understanding of the phenomena involved in the evaluation of translators' aids.

1. Translation and Evaluation – the Context

When evaluating translators' aids' systems, the first step should be the analysis of the context of translation. What types of tasks, problems and strategies are applied during translation, and where could these find support in computational tools; what types of tools are being developed to support the translator; and which are those that are most prominent on the translation market?

Similarly the approaches and efforts of evaluating translators' aids have to be considered. What are the most important attempts of evaluating translators' aids, what efforts have been made; what is the basic complexity of this undertaking; and where to look for the most pre-eminent evaluation research efforts.

1.1 The Translation Context

Lörscher (1991:5) and other scientists complain that the investigation of the translation process is still largely a desideratum. The most important reason for this is that there have been few attempts to assess the translation process in an empirical rather than theoretical-speculative way. Though the application of think-aloud protocols is the most often practised empirical method, doubts have been raised with respect to its adequacy to throw light on the entire translation process. According to Goguen/Linde (1993:157), normally, when translating a text, one does not talk aloud about one's ideas and thoughts and thus the situation as such is rather artificial. Hönig (1991:82) complains that subjects only verbalise the conscious part of their thoughts and moreover are only inclined to say what they are expected to. Also, the more professional translators become, the more translation becomes an automatic, unconscious process and the less they are inclined to verbalise their thoughts. Consequently think-aloud protocols report only part - and a highly selective part at that - of the mental processes. Kußmaul (1991:91) therefore proposes to use the method of *dialogue protocols*, where two people are asked to translate a text together and have to discuss their activities while they are getting on with the job, explain and justify their translation, make suggestions for improvement, ask for advice and criticism, all of which are features of natural discourse.

Since the way translation problems are perceived and solved differ from translator to translator, the first step in investigating the translation process is to apply the principles of generalisation and simplification in order to arrive at general phase-models of the translation process (chapter 1.1.1). Following this top-down approach the models are filled with empirical data concerning the actual tasks performed, the

problems encountered the strategies applied and the knowledge needed for solving these problems 1.1.2).

1.1.1 A Model of the Translation Process

There are two major types of models, which have been established in the course of the last three decades, that is, the two-phase model and the three-phase model. The major difference between the two models is not the actual number of steps or phases involved in the translation process but rather the fact that in the latter case, that is, the three-phase model, the transfer from the source into the target language is considered to be performed via a supralinguistic medium, whereas in the first case, source and target language units are considered to be directly correlated. There is no consensus among linguists as to which one comes closer to the "real" nature of translation.

Nord (1991), Wilss (1992) and many other translation scientists agree that translation cannot be considered a sequential process. This means that at each step forward the translator "looks back" on the factors already analysed, and that every piece of knowledge gained in the course of the process of analysis and comprehension may be confirmed or corrected by later findings. To accommodate this fact, Nord (1991:pp.30) developed the so called “looping model” which describes translating as a circular, basically recursive process comprising an indefinite number of feedback loops, in which it is possible and even advisable to return to earlier stages of the analysis. The "looping model" takes into account the textual and pragmatic background of the source and target texts. This model has been selected for the purpose of this study and the translation process will be viewed in the larger context of industrial documentation.

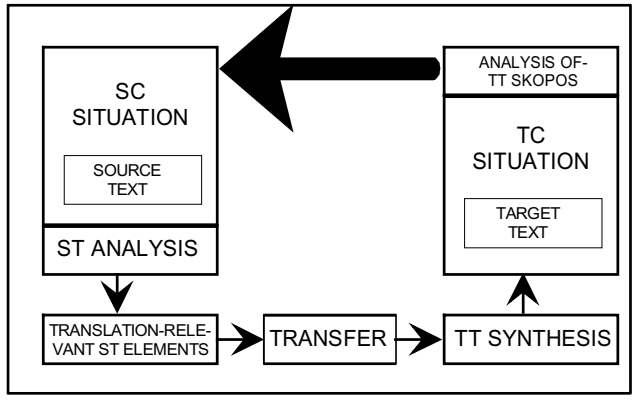


Figure 7: The Looping Model by Nord (1991:34)

As the above figure shows, Nord considers the first step in the translation process is to analyse which factors are relevant to the realisation of the eventual purpose of the target text (she calls this TT “skopos”). Nord identifies a set of "extratextual" and

"intratextual" factors that are the basis for the formulation of basic translation instructions. The following figure outlines the interplay between extra- and intratextual factors in text analysis, which Nord expresses by means of a set of "WH-questions".

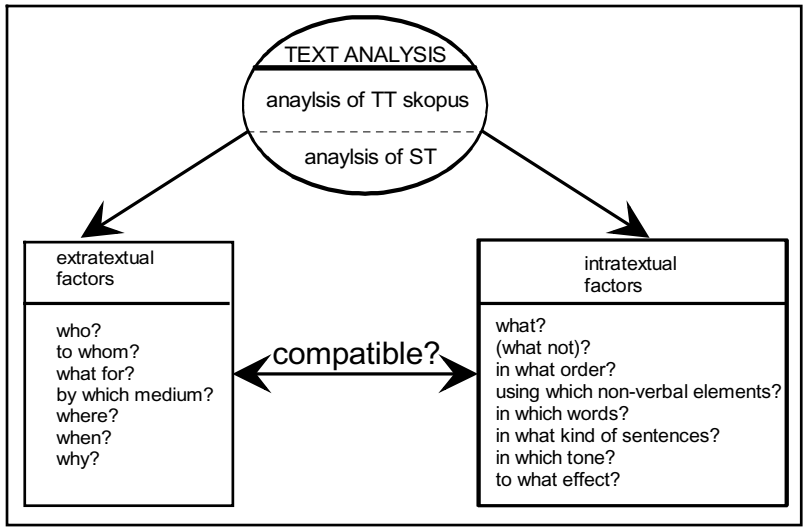


Figure 8: Text Analysis: "extratextual" and "intratextual" Factors by Nord (1991:36)

The second step in Nord's model starts with a rough analysis of the source text and its textual material, leading again to both a set of extratextual and of intratextual factors. First, the translator only has to get a general idea on whether the material provided by the source text is compatible with the translation instructions. However, Nord realises that both procedures of text analysis are in reality closely related and often have to be combined, demonstrating the recursive character of the model. Since the situation normally precedes textual communication and determines the use of intratextual procedures, it seems natural to start with the analysis of the external factors, although, in view of recursiveness and circularity, the order of the analytical steps is not a constituent of the model.

Whereas most linguists agree on the set of intratextual factors relevant to translation, there is no consensus considering the exact kind of extratextual information relevant to a translator. There are different information requirements, depending on whether translation of literature or of technical documents is concerned. Having performed a rough analysis of extratextual and intratextual factors, the next step in Nord's model is to comprehensively analyse all ranks of the source text, focusing on those elements that are of particular importance according to the purpose of the target text. In practical terms the translator might even during analysis of the source text come across questions which have something to do with the purpose of the target text and thus has to contact the client/author in order to get the missing piece of information.

After finishing the source text analysis, the translator is able to pinpoint the translation-relevant elements of the source text and adapt them to the purpose for which the target text is intended. In the transfer step, these elements are matched with corresponding target language elements, and the decision is made which of the potentially appropriate target language elements will be suitable for the target text function. The final structuring of the target text closes the circle. The translation is successful, if the target text is compatible with the pre-defined purpose of the target text.

1.1.2 On Specifying Tasks, Problems, Strategies and Knowledge Bases

The procedure developed in this thesis is to bring together the relevant facts and most enlightening ideas from various sources and to integrate them into the framework of the looping model briefly described above. Nord's text-oriented model will be further enriched with data from other translation theorists, data gained in empirical research and approaches from cognitive psychology. The combined model will be applied to the context of industrial documentation. It integrates experiences gained by the author of this thesis during a 5 years research contract in the translation department of Mercedes-Benz AG.

As has already been pointed out, the objective of this chapter is to investigate the extent to which tools could possibly be provided which assist the translator in his/her work. For this purpose a top-down approach will be adopted, starting from the above defined translation **phases**, considering the **tasks** involved, outlining the **problems** encountered, determining the **strategies** employed, and specifying the type of **knowledge base** tapped to solve the overall translation task. The following figure demonstrates the approach used in this thesis.

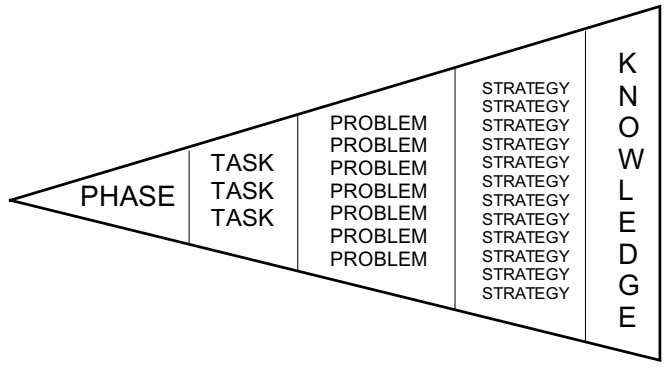


Figure 9: Top-down Approach for the Investigation of the Translation Process

Each translation **phase**, that is, analysis, transfer and synthesis consists of various **tasks**, which are procedures that are concerned with one type of work, e.g. the analysis of intratextual factors as opposed to the task of analysis of extratextual factors. During each task the translator has to solve different **problems**. The number of problems encountered depends strongly on the individual, that is, expertise, experience etc. comes into play. Since different **strategies** are applied by different individuals to solve translation problems, it is rather difficult to pinpoint one strategy to one particular problem. Attention will be paid to those strategies that are generally acknowledged. Whereas phases, tasks, problems and even strategies are by and large objectively observable units, it is difficult to determine how the internal processing of the data works and how the translator taps his/her **knowledge base**. Researchers influenced by cognitive psychology distinguish between two basic structures of memory - epistemic and heuristic memory (Dörner, 1979:27). Similarly Wilss (1988:82) distinguishes between "deklarativem" vs. "übersetzungsprozessualen Wissen" or (1992:115) between "statisches Sachverhaltswissen" vs. "dynamisches Wissen" or (1992:45) between "Akkomodation" vs."Assimilation". Hönig (1991:78f), distinguishes between "controlled workspace" and "uncontrolled" which roughly fall into the same two structures of memory. In short, the epistemic part of the memory or knowledge base determines the ability to solve problems reproductively, that is, to retrieve something which has been stored beforehand, or in other words, to access available expert knowledge. According to Wilss (1988:86) translators gradually build up an "encyclopaedic memory" which is first of all consulted during the translation process. Since access to the epistemic memory often functions automatically, such procedures are not necessarily reported in empirical investigations. Only if epistemic procedures are not successful, the translator has to apply his/her heuristic knowledge, has to be creative and stimulate cognitive resources and develop new problem solving methods. Thus Wilss (1988:86) argues that the more extensive the epistemic knowledge base, the less time has to be spent applying heuristic strategies. Experience not only builds up the epistemic memory but also trains heuristic knowledge strategies. Thus Sternberg (1984:283) concludes: "... experts are more competent in handling familiar tasks within the domain of expertise. They are also more proficient at learning new tasks, because global processing resources are more readily available for the intricacies of the task or situation confronted ... Experts are also able to perform more distinct kinds of tasks in parallel, because whereas the global system is conscious and serial in its processing, multiple local processing systems can operate in parallel."

On the basis of the above-presented facts, for each step an overview will be given which outlines the specific tasks involved. Attention will be focused on those tasks, which seem to have potential for automatisation. In this case the related problems and

strategies will be described. For each problem it will be discussed whether the translator is likely to have used epistemic or heuristic problem solving strategies. It is envisaged that the picture gained by means of this approach will provide an insight, which may help to determine programs and routines that could be implemented to ease and speed-up translation work.

1.1.2.1 Phase One: Text Analysis

Text analysis refers to both the analysis of the TT skopos as well as to the analysis of the ST. Both types of analyses have in common that they elicit information with respect to extratextual and intratextual factors. Since the two kinds of analyses are closely interrelated and, moreover, for the purpose of this study a strong distinction between the definition of the TT skopos and the analysis of the ST is not relevant, the following will only outline as task 1 the definition of extratextual factors and as task 2 the definition of intratextual factors.

The two major disciplines that consider extratextual factors are communication science and pragmatics. Nord (1991:39-70) elaborates the factors of medium/channel, place, and time of communication. The following figure follows Nord's investigation and provides an overview of sub-tasks undertaken when determining extratextual factors.

EXTRATEXTUAL FACTOR	PROBLEMS TO BE SOLVED
SENDER	• Who is the sender of the text? • Has the sender written the text himself? • What information about the sender can be obtained from the text? • What clues can be obtained from other situational factors?
SENDER'S INTENTION	• Are there any statements of intention? • What intention can be conveyed by the text type? • What clues can be obtained from other situational factors?
RECIPIENT	• What information can be obtained about the recipient? • What can be learned about the recipient form the information about the sender? • What clues can be obtained from other situational factors?
MEDIUM/ CHANNEL	• Has the text been taken from a written or spoken document? • By which medium is it transmitted? • What clues can be obtained from other situational factors?
PLACE	• Where was the text produced or transmitted? • Is any information about place presupposed to be part of the recipients' background knowledge? • What clues can be obtained from other situational factors?
TIME	• When was the text written? • Is any information about place presupposed to be part of the recipients' background knowledge? • What fundamental problems arise from a possible time lag between ST and TT situation? • What clues can be obtained from other situational factors?
MOTIVE	• Why was the text written or transmitted? • Is the ST recipient expected to be familiar with the motive? • Was the text written for a special occasion? • Is the text intended to be read or heard more than once/regularly? • What problems can arise from the difference between the motive for ST production and the motive for translation? • What clues can be obtained from other situational factors?

Figure 10: Elicitation of Extratextual Factors

Analogous to Nord's general questions, the following specific aspects have to be considered in industrial documentation:

- who is the author of the text (education/expertise/position)?
- who will be the target user of the document (education/economic situation/interest/age/sex etc.)?
- what is the overall message that has to come across?
- which will be the typical situation in which the target text will be received (medium/place/time)?
- which form/style will be most appropriate for the given combination of user/situation?
- which is the envisaged effect (emotional/practical) of the text on the user of the documentation?

These are among the aspects the translator has to consider either before or during reading/analysing the source text. The most promising source of information in the

industrial context is the client, that is, the translator has to contact either the author of the source text or even his/her manager in order to get a satisfactory picture of the envisaged TT situation.

Nord (1991:84-143) distinguishes between eight intratextual factors, that is, subject matter, content, presuppositions, text composition, non-verbal elements, sentence structure, and suprasegmental features. Whereas Nord concentrates on the question how to find information on the different intratextual factors, the following investigation will focus on the kind of problems the translator has to face on the level of intratextual factors. Moreover, since this chapter is mainly concerned with industrial documentation, Nord's model will be adjusted to this situation, that is, only relevant items will be discussed, additional considerations included.

Subject Matter and Content

Among all intratextual factors, subject matter and content are the most important ones for industrial documentation. The following figure summarises aspects of subject matter and content that are relevant in the context of this study.

ANALYSIS OF SUBJECT MATTER	PROBLEMS	STRATEGIES	KNOWLEDGE STRUCTURE
definition of subject matter?	coherent text ? (only one subject matter)	analysis at level of lexical items	HS
	text combination? (several subject matters)		
	hierarchy of subjects?		
isolation of information units	difficult syntactico-semantic structures?	paraphrase ST segments	HS
determination of extralinguistic reference	knowledge on subject matter available?	text documentation in SL	ES
	comprehension of individual concepts	terminology elaboration and look-up/text documentation - definitions - concept structures	ES

Figure 11: Analysis of Subject Matter in Industrial Documentation

The number of problems which are likely to occur with respect to the subject matter largely depends on the question whether the text covers one or more topics. In order to find out what the text is all about, the translator analyses the topic structure of a text and arrives at a network of semantic relations, which provide him/her with an overview of the overall subject matter. In case it is difficult to isolate information units in the ST, the respective parts are paraphrased in the SL. Within each topic the translator is faced with comprehension problems on the level of individual terms or on the level of subject knowledge. In these cases there is the need to clarify the

extralinguistic references to the lexical items and their relation to each other within the overall topic by means or text documentation or by considering definitions and concept structures of the relevant concepts. Whereas the definition of the subject matter and the isolation of information units are heuristic processes, the determination of extralinguistic references can be considered data-oriented and epistemic. The translator will first of all consult his/her "encyclopaedic memory" in order to solve the problem of extralinguistic reference. Only if this process is not successful, s/he will turn to the strategies described.

Presuppositions

The problem of presuppositions occurs when sender and receiver do not share the same background knowledge. In industrial translation a situation may occur, when the external reality of both sender and receiver are not congruent. In this case the translator has to apply heuristic problem solving strategies and add certain aspects of the external reality, which the receiver of the document is not likely to have access to. A typical example would be the translation of computer manuals for speech communities in which people are less computer literate, for instance, when selling Japanese computers in Cuba.

Text Composition

Text composition may pose some minor problems to the translator in industrial documentation on the macro and micro levels of the text. There may, for instance, be a composition specific macrostructure in the target language (e.g. for letters) that has to be noted in the translation instructions. Also different speech communities may have different conventions with respect to the microstructure of specific technical texts (e.g. whether complex or simple sentence structures are preferred). The translator would normally try to retrieve from his/her encyclopaedic memory (ES), how macro- and microstructures are defined.

Non-Verbal Elements

Non-verbal elements are signs taken from other, non-linguistic codes, which are used to supplement, illustrate, disambiguate, or intensify the message of the text. In industrial documentation there are a number of non-verbal elements to be found. Despite this frequency of occurrence, the probability that translation problems arise is rather low (not considering the aspects of form). This is due to the fact that the more technical the subject matter, the more standardised are the ways of expressing reality. Again the encyclopaedic memory is likely to have stored such aspects and if not, text documentation in both languages will help to solve the problem.

Lexis

The characteristics of lexis are considered in most approaches to the translation process. The choice of lexical features is strongly determined by extratextual factors. Thus aspects such as metaphors or repetition of lexical elements as well as certain rhetoric figures are of minor importance in the case of industrial documentation. However, sometimes there may be particular reasons for the choice of lexical units such as the archaic language for legal documents. To solve such a problem, the translator would normally consider texts of the same area, which again can be considered as epistemic rather than heuristic strategy. Another lexical problem which might occur is that certain lexical fields are used, that is, in-house terminology, metalanguages etc. The translator in this case would have to retrieve from epistemic memory or, if not successful, apply heuristic problem solving strategies to analyse the lexis and determine the translation instructions to be considered in the TT.

Sentence Structure

The sentence structure is likely to be different in industrial documentation than, for instance, in literature. The question of the density of terminology in individual sentences may arise, which would have to be analysed carefully (HS) to be later represented in the TT. Attention should also be paid to the question whether co-ordinated or sub-ordinated sentence structures are used and whether the same structure is likely to be used in the target language speech community. Again only heuristic procedures will lead to the goal.

Suprasegmental Features

In documentation suprasegmental features are signalled by optical means, such as font, italics, bold, quotation marks etc.. The translator is usually asked to translate form-neutral, so suprasegmental features have to be kept. It is the translator's heuristic knowledge that tells him/her, which elements have to be highlighted in the target language as compared to the suprasegmental features of the target language.

Having analysed the above intratextual and extratextual factors, the translator arrives at translation instructions, which pinpoint translation relevant elements and point out what has to be kept in mind during the transfer and synthesis phases.

1.1.2.2 Phase Two: Transfer

The most sophisticated account of what has to be considered in the transfer phase has been elaborated by Nida/Taber (1982: pp. 99-119) and Hohnhold (1990: pp. 35-95). For the purpose of this chapter two major tasks will be considered, that is, transfer of extralinguistic knowledge and semantic adjustment.

TASKS IN TRANSFER	PROBLEMS	STRATEGIES	KNOWLEDGE STRUCTURE
transfer of extralinguistic knowledge	TL terms belonging to a particular concept? (synonyms, variants, antonyms)	text documentation in TL;	ES
	set expressions or standardised terms?		
	concepts belonging together in TL? (relations, fields, systems)		
	special subject hierarchy available?		
	fixed scope of terms?	comparing SL and TL documents	HS/ES
	most suitable term for the TT skopos?		
semantic adjustment	idioms?	adjust to TL or paraphrase	HS
	formula?	adjust to TL	HS

Figure 12: Tasks Involved in Transfer

On the basis of the information elaborated by means of data-oriented epistemic means, the translator has to decide, which terms to use in the TL and whether the scope of the terms he considers appropriate suits the TT skopos - a procedure which is of heuristic nature. Finally semantic adjustments have to be considered (HS) if the SL or TL texts comprise idioms or formulas.

1.1.2.3 Phase Three: Synthesis

In the synthesis phase, translation units have to be combined into an acceptable TL text. In the following, two tasks will be distinguished, that is, the combination of translation units and the checking or evaluation of the translation against the pre-defined TT skopos.

TASKS IN SYNTHESIS	PROBLEMS	STRATEGIES	KNOWLEDGE STRUCTURE
combination of translation units	syntax OK?	consider context of terms in other TL documents	HS/ES
	morphology OK? irregularities?	consult dictionaries text documentation in TL	HS/ES
	technical collocations to be considered?	text documentation in TL	ES
	phrasing conventions?		
evaluation of translation against TT skopos	TL text congruent with TT skopos?	comparison of features mentioned in TT skopos with features of the text	HS

Figure 13: Tasks Involved in Synthesis

The phases, tasks, problems, strategies and knowledge structure elaborated above are by no means meant to be exhaustive. In some cases, both knowledge types were referred to, because a combination seemed likely. The strategies applied by translators and the type of knowledge applied to solve problems will form input into the development of metrics that measure the quality of different translators'aids' systems. Those systems that support strategies that are typically applied by translators to solve the various translation problems discussed, will rank high in the overall assessment procedure.

1.1.3 The Professional Context of Translation

The above discussion of the translation process is meant as a step towards the more detailed description of problems and strategies involved in translation with a focus on industrial translation as opposed to the translation of literature. What the requirements of translators as users of translation tools eventually are, however, largely depends on the professional context in which the translation job is located. In other words, what is needed depends to a certain extent on (i) how many people work together; (ii) which resources they can share; (iii) and how they divide up the work. Along these criteria it is useful to distinguish between two major translation contexts, that is, freelance translating and midsize to large translation contexts. The following two sections take into consideration Fulford/Höge/Ahmad (1990); EAGLES (1995) and observations of the author during her 5 year research work at the translation department of Mercedes-Benz.

1.1.3.1 Freelance Translating

What characterises freelance translators as opposed to translators working alongside other translators in midsize translation companies or in-house translation departments, is above all the variety of text types and subject areas they have to deal with. Due to this immense complexity, nowadays many freelancers, if they can afford it, tend to specialize in only a few subject areas. The type of translation work freelancers usually undertake is translating, and only few are also proofreading or revising target language texts. Most freelance translators have between five and 10 clients they regularly work for. They translate hundreds of pages a year into a single language direction. Consequently freelancers cannot fall back on large specialized resources of either terminology or translated texts. Freelancers equally have to deal with administrative, translation and technical problems involved in the context of the translation job.

1.1.3.2 Midsize to Large In-house Translation Departments

Translators in midsize to large in-house translation departments occupy a specific role in the overall process of documentation. Consequently industrial translation must not be considered as a discipline in its own right but rather as the final stage of the larger documentation process. The way the process of documentation is performed, determines the responsibilities of the individual translator. The documentation process, was formerly characterised by paper as the primary medium. It involved a number of time-consuming loops in which the documents had to be composed, checked and printed by third parties. The mere task of the translator then was to transform a text from one language into another and to try to fit the translations into the given galley proofs.

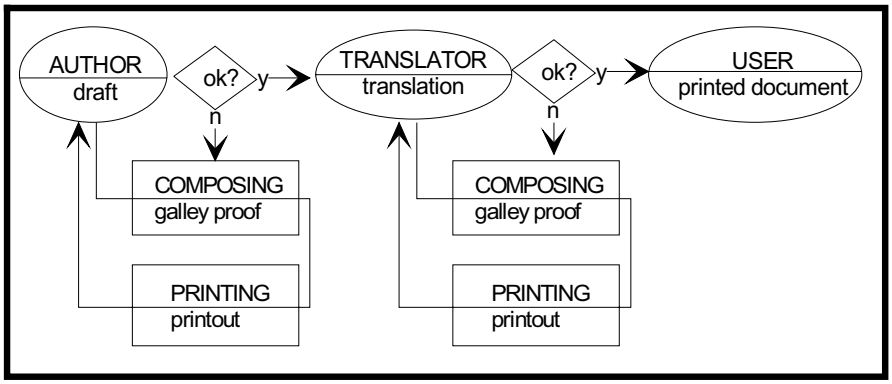


Figure 14: The "off-line" Process of Industrial Documentation by Höge/Kroupa (1991:1037)

Within the last decade, however, the introduction of new technologies have led to a considerable change in the overall process of documentation (Höge/Hohmann/Le-Hong 1995: 4). This has gradually resulted in an increase of responsibilities on the side of both authors and translators. Authors had to start thinking about formal matters of presentation and similarly the translator has become to be responsible for the final shape of the TL documents, taking over most of the composing and printing jobs formerly performed by different people.

Moreover, with shortening product life cycles, there is a strong demand that the document life cycle be reduced to its minimum, while at the same time the volume of technical documents is increasing (more products, more documentation). In medium to large translation departments, the amount of texts that need to be translated varies from one to tens of thousands of pages a year (cf. EAGLES, 1994:141). This situation puts great demands on authors and translators equally. Both are asked to develop strategies

to ease and to speed up their work, resulting in a change of both responsibilities and workflow.

On the author side, two main strategies could be observed which reduces text production effort, that is, standardisation and re-use of text. Investigations on the repetitiveness of certain text types by Fulford/Höge/Ahmad (1990:17) showed that it is above all technical documentation that bears a great amount of repetitions both within the same document, within documents of the same type or even other versions of the document. The principal assumption for this kind of investigation was that repetitions are only the rule in less sophisticated text types such as manuals. However, a new trend in documentation is gradually emerging: authors, generally under time-pressure, also started to compile stylistically sophisticated text modules for the description of products or parts thereof in public relation documents. Instead of reformulating the same facts about a particular product (for instance, a new turbo diesel engine), authors now combine a great number of such modules within one overall text. This way an enormous amount of time can be saved without decreasing the stylistic level of the documents.

On the translation side, however, strategies to decrease translation work are only gradually being employed. The current situation for most translators is still that they receive their documents either as a printout or on a floppy, work with their text processors, consult dictionaries (maybe on-line dictionaries), card files, magazines and alike and finally produce their target texts. Many in-house translators realise during translation that there is "a good deal of repetition" in the texts they are working on, but when being asked to quantify repetitiveness in documents, they cannot give more than a general impression, and decide that it is less time-consuming and less effort to re-translate passages instead of looking for previously translated texts. Accordingly, an important task in the analysis of the source text is to determine the quantity of actual translation work, that is, to watch out for recurring passages of text within the same document or within other, already translated documents.

Translators in midsize to large translation departments are usually highly specialized both in terms of subject area and text type. While they tend not to be responsible for either administrative or technical problems involved in the translation process, the type of translation work has changed from mere translation to translation and layout. Moreover, having to deal with large texts such as car manuals, one translator is often only responsible for a specific part of the text. Problems of the consistency of terminology and, probably, also of style have to be dealt with. In multinational companies the extent of translation work also frequently covers proofreading and

revision of foreign language texts as well as the preparation of terminology for use throughout the international company.

To conclude, from the above description of typical professional contexts of translators it follows that there are major differences between freelancers and midsize to large translation departments with respect to

(i) the type of tasks tackled during translation;
(ii) the workflow, that is, the sequence of tasks; and
(iii) the importance of the different tasks

On the basis of the above criteria, typical user profiles of translators may be drawn up, which represent the general requirements of these major two professional working contexts.

1.1.4 Candidates for Automation

So far, the basic issue concerned with the motivation of this chapter has not been tackled, that is, why should new translation tools be developed or existing ones improved? The major reason for introducing translation tools is that every profession has to keep pace with technology to guarantee competitiveness. Thus, if generally all product life cycles are getting shorter and the factor "time" is of increasing importance, the translator, too, should seek to increase the quantity of his/her work while keeping the quality at least at the same level. This, however, cannot be achieved without tools that support various tasks in the translation process.

The following approach is meant to bridge the gap between the theory of translation and theoretically possible translation tools. Wilss (1992:105) points out that there is some basic parallel between the human brain as information processing device and the computer. Both "systems" process information that has somehow been entered, is stored in the memory and that can be retrieved by different operative search strategies. Considering the tasks, which have been defined and elaborated above, the main interest will lie in the distribution of the types of knowledge involved when solving translation problems, that is, epistemic, data-oriented versus heuristic, process-oriented knowledge. The interpretation of the data will be based on the hypothesis, that for epistemic, data-oriented knowledge there is a high potential of automation, whereas heuristic processes are difficult to be described and therefore, at the current state of technology, their automation is only gradually being investigated.

The following table gives an overview on the distribution of knowledge types among the problems discussed above. According to the figures, there is an equal distribution between the application of epistemic and heuristic problem solving strategies even in

the context of industrial documentation. One may assume, that for literature translation, the percentage of heuristic structures will be much higher than for technical documentation, where the extralinguistic reality is more likely to be similar between sender and recipient.

TYPE OF KNOWLEDGE INVOLVED	NUMBER OF PROBLEMS DISCUSSED
epistemic knowledge (ES)	12
heuristic knowledge (HS)	12
heuristic and epistemic (HS/ES)	4
total of problems discussed	28

Figure 15: Distribution of Epistemic and Heuristic Knowledge Structures

Future research in the area of both heuristic translation strategies and the emerging technical discipline of *neural networking* may result in a better understanding of the matter and, in the long run, even lead to the development of tools that support heuristic translation strategies. Currently, even state of the art technology, however, mainly allows that problems and strategies that involve epistemic knowledge are supported by the computer. Consequently, the more intelligent tasks, such as reflecting, considering, and deciding are still mainly performed by the human translator. However, if Wilss' (1988:86) assumption is true, that is, that the more epistemic knowledge is available during the translation process, the fewer translators have to perform heuristic processes, the introduction of efficient tools based on epistemic knowledge, would nevertheless decrease heuristic activities of human translators. The following table is a summary of those previously discussed problems that are solved by the aid of epistemic strategies.

PHASE	TASK	PROBLEM	STRATEGY
ST analysis	determination of the quantity of the actual translation task	repetitions within the document or already translated documents?	compare sentences within the same document or the document with existing text corpus
	determination of extralinguistic reference	knowledge on subject matter available?	text documentation in SL
		comprehension of individual concepts?	terminology elaboration and look-up/text documentation SL - definitions - concept structures
	text composition	composition specific macrostructure?	text documentation/ parallel text[1]
		different conventions with respect to the microstructure?	text documentation/ parallel text
	non-verbal elements	usage of non-verbal elements conventionally bound ?	text documentation/ parallel text
	lexic	particular reasons for the choice of lexical units?	parallel text
transfer	transfer of extralinguistic knowledge	TL terms belonging to a particular concept? (synonyms, variants, antonyms)	text documentation in TL/
		set expressions or standardised terms?	text documentation in TL/
		concepts belonging together in TL? (relations, fields, systems)	text documentation in TL/
		special subject hierarchy available?	text documentation in TL/
TT synthesis	combination of translation units	technical collocations to be considered?	text documentation in TL
		phrasing conventions?	text documentation in TL

Figure 16: ES: Phases, Task, Problems and Strategies

Considering the above table, it is rather striking that all of the strategies employed are based on the use of text corpora. No matter whether for analysis, transfer or synthesis, text corpora are the adequate means to support the translator during all steps of the translation process.

Apart from straightforward epistemic aspects, it is interesting to consider those problems briefly that were classified as a mixture between epistemic and heuristic (HS/ES). As the following table shows, these problems generally involve a number of heuristic processes that have to end with the decision on the part of the translator (HS),

[1] note: in the translation theory context the term *parallel text* denotes texts of the same type in the source language (e.g. other legal documents); in the translation system development context, *parallel text* is often referred to as translations of a text, e.g. a parallel text corpus is a text corpus that includes source language texts and their translations.

which can mainly be made on the basis of either the availability of a certain amount of data (ES) or simple mathematical operations.

PHASE	TASK	PROBLEM	STRATEGY
ST analysis	investigation of sentence structure	density of terminology within a sentence?	identify terms in ST count number of terms in ST represent same density of terminology in TT
transfer	transfer of extralinguistic knowledge	fixed scope of terms?	define concept of terms in ST consider concepts in TL/ text documentation in TL decide which concepts suit best for given text/situation
		most suitable TL term?	
TT synthesis	combination of translation units	syntax OK? irregularities?	apply general syntax rules consider context of terms in TL/ text documentation in TL
		grammar OK? irregularities?	apply general grammar rules for irregularities consult dictionaries or text documentation in TL

Figure 17: HS/ES: Phases, Task, Problems and Strategies

The investigation of the sentence structure is a classical combination of applying heuristic processes in finding out, which of the elements of a sentence can be classified as *term*. Though this process is characterised by a number of complex sub-processes (e.g. consider lexical, semantic, syntagmatic aspects etc.) one may nevertheless imagine that one may arrive at similar results (here the determination of terms) on the basis of other, more algorithmic, processes such as the comparison of the SL text with an LGP (Language for General Purposes) data corpus. The results of such alternative processes would still have to be checked by a human translator and the final decision would remain in his/her hands. In addition to the rather complex problem of determining terminology, other sub-processes involve the simple counting of terms, which again can be defined as algorithm.

Determining the scope of a term involves both the definition of extratextual factors (situation etc.) which is the result of heuristic investigations, and the simple looking-up of the terms in TL text documents. The final decision about which term suits best in the TT has to be made by the translator.

The combination of translation units can also be considered a classical problem. It involves the application of rules (syntax, grammar) but also has to consider those

aspects that are not covered by rules, that is, irregularities in grammar or complementation. Again, the multilingual text corpus provides data for solving the problem, determining irregularities.

Taking into account all aspects mentioned in both the ES and HS/ES context, it becomes obvious that the potential to automate a great part of the tasks is striking. It goes far beyond the scope of this chapter to build up a detailed specification for a tool that integrates all aspects elaborated above. Nevertheless, a rough outline of the functionality of such a text corpus tool will be given in the following figure.

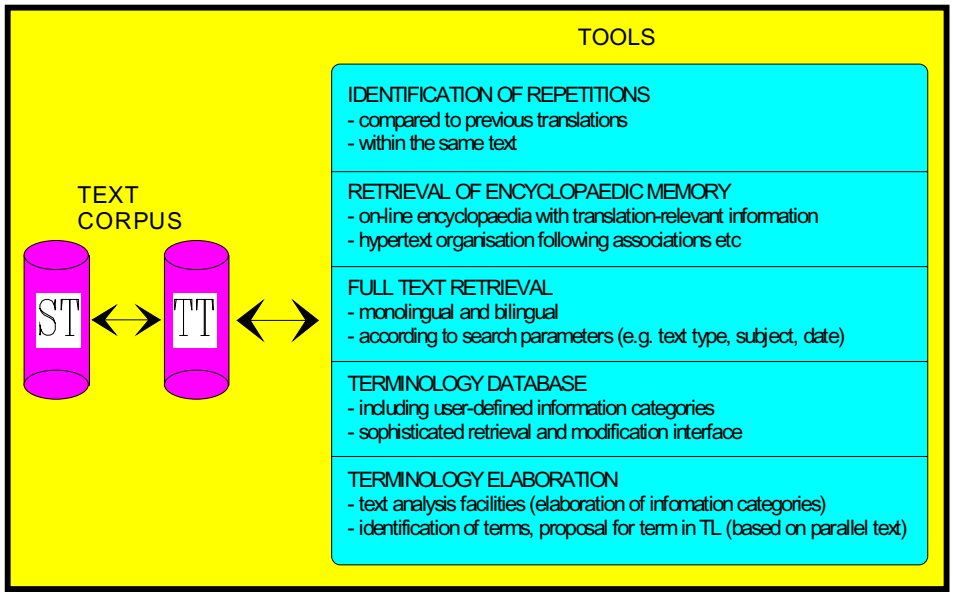


Figure 18: Possible Functionality of Multilingual Text Corpus Tool

The above figure shows that there is no reason why the translator should be confronted with different tools, comprising independent engines that make use of independent data resources. The translator is in the end always confronted with text and has to apply different operations on this text in order to arrive at his result, the "correct" translation. Thus it must be the first goal to provide the translator with tools to handle multilingual text corpora and to apply operations on the corpora - either automatically or "manually". A toolbox approach would make sense, in which translators "plug in" those tools which are of interest to their particular translation environment.

The above developed model of an “ideal” multilingual text corpus tool was based on the analysis of translation tasks. The following section presents an overview of types

of translation tools that are actually being developed both in the research and industrial context.

1.1.5 Types of Translators' Aids

Within the last decade a great number of translators' aids products surfaced, many of them to fall back into oblivion again after a short while. Which of the software products on the market are actually successful largely depends on the functionality of the modules they offer. The modules that are being developed in the industrial and research context cover multilingual dictionaries, multilingual thesauri, terminology management databases, translation memories, text alignment tools, and, last but not least, terminology elaboration tools.

Multilingual dictionaries typically consist of one or more alphabetically organised bilingual dictionaries and the corresponding retrieval software. Most retrieval software systems allow access to several dictionaries, though only one can be active at any one time. There are some computational systems that can assist dictionary editors to build up user dictionaries, which are then handled in a similar manner to the read-only dictionaries provided. Inflectional morphology look-up, where the input word is morphologically reduced to its word stem, is a rare but welcome feature of the tools currently on the market. More sophisticated packages offer add-ons for merging, inverting, importing and exporting dictionaries.

Multilingual thesarui are relative newcomers among translation software. They offer two or more monolingual thesauri, cross-referenced by concepts rather than alphabetically. By means of these links, the user can rapidly browse through the subject hierarchies in different languages. Multilingual thesauri in particular support written language and are interesting for all those who have to translate into or write in the foreign language.

Terminology management databases typically consist of a terminology database together with retrieval and modification software, which allows users to access and enter data in different background databases. The underlying term model has been the subject of ongoing debate for years. They are either term or concept-oriented and in most cases only offer a pre-defined set of information categories, such as grammar, definitions, context and the like. If at all possible, it needs considerable effort to adapt the given set of information categories to particular user needs. As in multilingual dictionaries, inflectional morphology in look-up routines is rare. Most systems currently offer utilities for maintaining termbases such as reversing, merging,

exporting, importing and printing facilities. A new feature of some more sophisticated systems is the support of terminology extraction from existing texts.

The concept of *translation memories* (TM) has been around for more than twenty years now but has only recently become an important commercial entity. Basically a translation memory makes use of already existing translations in that it tries to match a new source text - sentences or parts thereof - with existing source texts. Some TMs can only retrieve exact matches, whilst others apply fuzzy matching algorithms that allow the retrieval of near matches, for which the TM then automatically offers the translations of the previous versions. The quality of the matching algorithm and the underlying way of storing the data largely determine the performance of the systems. TMs are typically integrated into translation workstations and used in conjunction with terminology databases/multilingual dictionaries or even MT (machine translation) raw output.

In conjunction with TMs, more sophisticated systems offer *text alignment* tools, the results of which can subsequently be imported into a translation memory system. Thus the TM can make use of existing translations that originally were **not** translated by the aid of the TM. Central problems to text alignment tools are (i) the segmentation of the source and target language texts into translation units, and (ii) the alignment between the corresponding source and target language translation units. The closer translations are to the original, the higher the quality of the output of text alignment tools.

Recent developments include *terminology extraction tools.* Terminology extraction techniques and associated tools have been developed that can assist in the elaboration of terminology at all stages in such a process: from the identification of existing or emerging terms, through locating terms within a terminological hierarchy and finding their associated collateral, to the validation of terms. There are few tools or techniques that support all stages of terminology elaboration, and fewer still that consider the importance of managing the resources from which the terminological evidence is being "mined" or "discovered". The techniques that have been applied within this area come from two main approaches: statistical or semantic. The statistical approach comes from a line of research that goes back to the 19th Century and can be found today within the discipline of Corpus Linguistics. The focus of the statistical approach is to use concordance analysis, that is, the quantitative, numerical analysis of the occurrence of words. In contrast, semantic approaches typically apply deep linguistic analysis on smaller quantities of data to achieve their goals.

Since there is much progress in both hard/software and language engineering research, a classification such as the one presented above cannot be more than a cursory snapshot of a rapidly moving target. However, the classification showed that there is a certain overlap between the model of the ideal multilingual text corpus tool and what is actually being developed. Thus the identification of repetitions is realized in form of translation memory systems and terminology databases as well as elaboration tools are actually developed. The retrieval of encyclopaedic memory, as was proposed in the text corpus tool model, is only realized on a rather low level in the form of multilingual thesauri, and the full text retrieval module is still a desideratum.

1.1.6 Translators' aids Products on the Market

Considering the following survey, it is important to note that only the key players of the 1999/2000 translation market are briefly described. There are many more systems that offer to some extent language engineering facilities that are geared to the non-professional multilingual market, which, however, will not be considered further here.

1.1.6.1 TRADOS 5 Translators Workbench[1]

The *Translators'* Workbench offers a comprehensive translation solution, which can be integrated into Microsoft Word for Windows. TRADOS 5 is XML based and offers two product lines, that is, Freelance Edition and Team Edition. Central elements are the *terminology database* Multiterm® and a *translation memory* system which retrieves already translated segments with their translations from pre-stored databases. It produces fuzzy matches for all segments that are not identical but similar to those stored in the database. The "concordance" function further allows scanning the database for a source sentence, or a part of a source sentence and retrieves the corresponding matches. The user does not have to leave the normal text-processing environment when doing the translation, but additional windows are opened for the source language text, the target language text and the terminology database information. For each translated segment, the translations are stored in translation memory databases from where they can be retrieved during the very translation process or accessed for later translations. The workbench offers many options to customise the environment and therefore improve the output of the system. It supports the management of large translation projects, allowing the analysis and calculation of effort. Already existing translations can be aligned and imported into translation memory databases using the WinAlign® program, which visually supports the

[1] For more information see http:/www.trados.com.

alignment process. A range of tuning options provides precise control over the alignment process. The workbench is open for other applications such as Machine Translation Systems like SYSTRAN or Logos. The TRADOS S-Tagger offers interfaces to other DTP applications like FrameMaker and Interleaf. The TagEditor allows the translation of HTML/SGML texts, offering the full workbench functionality and user-friendly editing facilities. With TRADOS 5 three new features have been implemented:

(1) WorkSpace®, a workflow management software that integrates all TRADOS functions into an overall interface;
(2) Extraterm®, a statistically based terminology extraction tool, which offers possible translations to terms in a source language on the basis of an existing bilingual text in TRADOS format or a translation memory.
(3) Xtranslate®, a reference file based tool for update translations, which finds translations of segments in the same textual context and, therefore, reduces quality checking to a minimum.

The T-Window Collection, furthermore, includes a translation environment for Microsolf PowerPoint®, Exel®, clipboard and executable files. Multiterm IX®, finally, is a client-server based architecture for the distribution of terminology via intra or internet. The TRADOS products are supported throughout the world with thirteen offices in ten countries on three continents.

Supported Languages

- all Windows 95, 98 and NT supported languages (soon also including Arabic and Hebrew);

Supported Formats

- all popular formats: WordDoc, RTF, Online-Help RTF, PowerPoint, FrameMaker, FrameMaker+SGML, FrameBuilder, Interleaf, Ventura, QuarkXPress, PageMaker, SGML/HTML, RC (Windows Resource), Bookmaster (DCF), Troff.

System Requirements

- operating systems: Windows 95, 98, 2000 and NT.
- hardware environment: PC with Intel Processor and alt least 64 MB RAM
- networking environment: all popular environments such as Windows NT, Novell Netware, IBM LAN Manager.

1.1.6.2 STAR Translation Technology[1]

With Transit 3.0, Star Translation Technology offers a multilingual publishing solution with three major components, that is, Transit, the translation environment including a reference-based translation memory and alignment tool; TermStar, the terminology database for local and local area network applications, and Webterm, the terminology database for Inter- or Intranet applications. It also offers speech recognition technology that facilitates speech input. Already translated documents serve as the basis for the translation memory system. Before starting the translation all relevant files are analysed and translation segments identified. The translation editor can adopt the look-and-feel of Microsoft Word with the additional functionality of translation memory (including fuzzy matching) and TermStar, as well as notice board and project management facilities. For each translation segment, a total or fuzzy match is searched in the translation memory files and the current document. Terminology that can be found in the TermStar database is offered in a separate window. The Star Translation environment offers various possibilities for customisation for specific users or projects as well as project analysis functions calculating how much of a text could be pre-translated and to which extent. On the market are the following different scale Star Translation products: Transit and TermStar Professional 3.0, which are extensive networking solutions including project management facilities; Transit and TermStar Workstation 3.0, which do not include export/import facilities, TermStar Viewstation 3.0, which has read-only access to TermStar; and Smart 3.0, the local translation solution including Transit, TermStar and project management facilities.

Supported languages

- Chinese, Japanese, Korean, Thai, Indonesian, Vietnamese, Arabic, Turkish, Danish, German, English, Finnish, French, Greek, Italian, Catalan, Dutch, Norwegian, Portuguese, Swedish, Spanish, Bulgarian, Croatic, Polish, Romanian, Russian, Serbic, Slovakian, Slovenian, Chechic, Ukrain, Hungarian as well as language variants such as British and American English.

Supported formats

- Text and DTP formats: Microsoft Excel, ~ PowerPoint, ~Word, Adobe PageMaker,~ FrameMaker, Corel Word Perfect, Interleaf, QuarkXPress, configuration possibility for ASCII or ANSI based file formats
- Software localisation: Windows Help files (RTF), Windows Resource Files, C/C++ source code, Java source code, source code of other programming languages, SPS programs

[1] For more information see http:/www.star-group.net

- Generic Data Formats: SGML, HTML, XML

System Requirements
- operating systems: Windows 95, 98, 2000 and NT
- hardware environment: PC with Intel Pentium CPU, 133MHz or higher; and at least 32 MB RAM for Windows 95; 48 RAM for Windows 98, 2000 and NT; 80 MB hard disk, CD-ROM drive, SVGA Graphic card (800x600)

1.1.6.3 IBM Translation Manager[1]

The Translation Manager 2.6 offers a large-scale translation solution that is built up on a specific translation editor, which integrates both dictionary look-up and translation memory functionality. It can be used as stand-alone or networking application on an OS/2 or Windows platform. It makes use of a folder concept combining documents, translation memories and dictionaries including as many files as required in any format into folders. All relevant source language texts are located in a translation folder and in a first step analysed prior to translation. The analysis process defines translation segments, identifies matches and near matches, and terms in the dictionary. The translator proceeds segment by segment in the translation editor and is presented the results of the analysis process in different windows, that is, one for the SL segment, one for the matched translation segment and one for terminology. For each translated segment, the translations are stored in translation memory databases from where they can be retrieved during the very translation process or accessed for later translations. Translation memory databases can be created from existing translations with visual support for the alignment process. The Translation Manager allows the creation and update of terminology lists, offers various statistic project management facilities such as for word count, progress, or repetitions and offers interfaces for Machine Translation programs.

Supported languages
- Spell checking and full language support for more than 30 languages, including Eastern European, DBCS and BIDI languages;

Supported formats
- SGML/HTML, MS Word, RTF, Word Perfect, AmiPro, FrameMaker, FrameMaker, Interleaf, Ventura, QuarkXPress, PageMaker, MRI

System Requirements

[1] For more information see http:/www.ibm.com/software/ad/translat

- operating systems: OS/2 Warp Version 3.0 or higher; Windows 95, 98, and NT
- hardware environment: PC with 486/66MHz or higher; and at least 32 MB RAM; 200 MB hard disk, CD-ROM drive

1.1.6.4 Atril Déjà Vu[1]

Déjà Vu stores translations in a "memory database" and reads over the source text, instantly retrieving applicable translations whenever it finds something equal or similar to the SL sentence. Déjà Vu includes TermWatch, a fully integrated terminology management system, as well as a File Alignment Wizard that helps to create memory databases from existing translations. It offers additional functions such as Pretranslate, which inserts both any exact matches found and fuzzy matches as suggestions in the correct places; or Assemble, which takes a closer look at the memory database than Pretranslate, and in many cases can put together a translation out of pieces that were not sufficient for Pretranslate. It further allows scanning the database for a source sentence, or a part of a source sentence. Words in the TermWatch terminology database can be looked up and, if not available, the Learn function evaluates source and target sentences in the memory database to give the most probable translation for an unresolved term in the lexicon, or for an untranslated expression in the source text. Project management facilities count words and characters per file or per project, and allow analysing a project's internal repetition factor. They also support the completion of large multifile and multilingual translation projects.

Supported languages

- Spell checking facilities are included for US English, UK English, German, Spanish, Finnish, French, Italian, Dutch, Danish, Brazilian Portuguese, Norwegian (Bokmal) and Swedish. New dictionaries can be created;

Supported formats

- Word (including Word 2000), RTF, Help Contents, PowerPoint, FrameMaker, PageMaker, QuarkXPress, Interleaf, Java Properties files, HTML (including ASP), HTML Help, SGML, RC, C/C++, IBM TM, Trados Workbench, and plain text files

System Requirements

- operating systems: Windows 95, 98, and NT

hardware environment: Pentium processor with 32 Mb RAM is recommended

[1] For more information see http:/www.atril.com

1.1.7 Conclusion to the Translation Context

The first part of this chapter has gradually led us from translation theory and practice, resulting in a model of the ideal multilingual text corpus tool, over the types of translators' aids in development, to the actual products on the market. In short, on the basis of both the detailed analysis of the translation process and the consideration of the practical translation context, the development of translators'aids' systems could be justified. It became obvious that what is being developed and what is currently on the translation market does, to a large extent, correspond to what can be considered useful.

Of the products currently on the market, the TRADOS TWB is the most prominent, though the description of the systems on the market showed that in terms of general functionality there are no big differences between the competitors. So, one may ask, what is it that makes one system superior to the others? The answer to this question lies in the needs of specific users, the performance of the systems, and the assessment of this performance with respect to those needs – the basic tasks of evaluation.

1.2 The Evaluation Context

The evaluation of translators' aids has been an acknowledged necessity for several decades now. Particularly the evaluation of machine translation systems received some attention as early as in the nineteen sixties. The ALPAC report (1966) was one of the first evaluation reports that attempted to measure the adequacy of machine translation in terms of informativeness and fidelity. Though the measures used were both ill-defined and merely based on subjective judgements, the committee's extremely negative conclusions about what machine translation could achieve in the short to medium term influenced the funding of machine translation research for a considerable time. Since then there have been numerous evaluations of machine translation systems, many of them on behalf of potential customers or in the context of translation teaching. Representative examples are the evaluations of SYSTRAN in CETIL (1979) or Heid (1990); and METAL by Slocum et al. (1985), JEIDA (1992); ARPA (1994). A comprehensive review of machine translation and evaluation efforts can be found in King (1984) and Falkedal (1991).

More recently evaluation has concentrated on translators' aids' systems rather than on machine translation. Again there are many attempts to evaluate these systems (i) in the context of translation systems research and development; (ii) on behalf of potential customers or (iii) in translation teaching. Examples for this type of evaluation are: WHA (1993); Spies (1995); Schüller (1995); Reinke (1994). However, what all of

these evaluations have in common is that they consider specific evaluation problems only or present a mere comparison of features of translators' aids.

The necessity to develop a methodology for the evaluation of translators' aids' systems has been repeatedly discussed and so far a number of attempts have been undertaken to perform methodically informed evaluation of translators' aids' systems. Thompson (1991 and 1912) are typical examples of theoretically driven efforts towards evaluation methodology. One of the first practically driven efforts to develop an evaluation methodology for translators' aids was performed in the context of the Translators'Workbench Projects (2315 and 6005) within the ESPRIT II framework edited by Kugler/Ahmad/Thurmair in 1995 or described by Höge/Hohmann/Le-Hong (1995).

Efforts have been made to adjust and apply evaluation methodologies that were already established in the fields of other natural language processing areas such as in database queries or fact extraction performed by Chinchor (1991) or Flickinger et al. (1987).

There are a number of initiatives that have taken up the evaluation topic at large. One of the most important attempts to produce a framework for the evaluation and assessment of natural language processing systems was undertaken in the EAGLES evaluation and assessment group, which was called into life by the European Union. EAGLES was the basis for several other initiatives such as ELRA, ISLE, DiET, or ELSE. In the following, EAGLES and its successors will be described briefly in order to gain an overview of the work that has been performed in this area.

1.2.1 EAGLES Evaluation Group

A prerequisite for the advent of language engineering technologies is the availability of a basic infrastructure comprising reusable linguistic resources, specifications for standards and related software tools - the objectives of EAGLES.[1] Part of this thesis, specifically the test types which will be described in more detail in chapter 4, went into the EAGLES Evaluation Group final report.[2]

EAGLES was formally established in January 1993. With a Community funding of around 1.25 Million ECUs the group intended to draw up a set of language engineering guidelines in 1995. It was split into two operational phases of 15 months

[1] http://www.ilc.pi.cnr.it/EAGLES/home.html
[2] http://issco-www.unige.ch/ewg95/ewg95.html

each and had the active participation of more than 30 research centres, industrial organisations, associations and research networks covering most EU countries.

EAGLES was intended to respond to the lack of common technologies and standards for the language industries. From a practical point of view, the major objectives of EAGLES were

(i) to produce agreed specifications and guidelines for specific areas of language engineering and make recommendations for a more uniform approach; and

(ii) to bring together the different approaches of industry and academia and foster their collaboration. Five main areas were identified to form working groups:
   (1) Text Corpora
   (2) Computational Lexicons
   (3) Linguistic Formalisms
   (4) Evaluation and Assessment
   (5) Spoken Language Resources and Methods

The working group on evaluation and assessment was split up into three subgroups, that is, Writer's Aids, Translators' aids and Information Management Systems. Jointly the three groups strove to set up guidelines for the evaluation of language engineering products and to exemplify the validity of the guidelines by applying them in the three areas of interest. The Evaluation Group managed to bring together the principal concepts of evaluation of language engineering systems and the experiences made in different areas within a number of EU projects.

According to EAGLES one of the principal questions in evaluation is to define what evaluation is for, that is, is an evaluation intended as means of demonstrating scientific merit, or of determining commercial viability, or of aiding development, and related to this, who evaluation is for. EAGLES argued that depending on both the intention behind the evaluation exercise and the target user of the evaluation results, different procedures are likely to be applied (Galliers/Sparck Jones, 1993:139). EAGLES distinguished between three types of evaluation, that is

- **adequacy** evaluation: the activity of assessing the adequacy of a system with respect to some intended use of that system.
- **progress** evaluation: the activity of assessing the actual state of a system with respect to some desired state of the same system.
- **diagnostic** evaluation: the activity of assessing the state of a system with the intention of discovering where it fails and why.

Due to time and budget restrictions, the efforts to develop a methodology for evaluation was restricted to adequacy evaluation. An example of adequacy evaluation is when a potential customer investigates whether a system, either in its current state or after modification, will do what he requires, how well it will do it and at what cost. Or, in other words, adequacy evaluation involves a pre-defined set of needs and evaluates a system's ability to fulfil those needs. It can be compared to the kind of evaluations, which are performed by consumer organisations for cars, washing machines, hardware, software etc. This Consumer Report Paradigm was a central precept for the elaboration of the evaluation framework for adequacy evaluation in EAGLES. The EAGLES final report, however, shows that despite all efforts, no definite methods could be found that led to the assessment of translators' aids' systems in terms of user needs.

1.2.2 ELRA and Evaluation

ELRA (European Language Resources Association)[1] was established in Luxembourg in February, 1995, with the goal of founding an organization to promote the creation, verification, and distribution of language resources in Europe. A non-profit organization, ELRA aims to serve as a focal point for information related to language resources in Europe. It is also concerned with the validation of language resources as well as of machine translation evaluation. Before distribution can proceed, for everything except research use, the product must be subject to quality control and validation. In the first place, the development or research project must draw up a manual for validation, and persuade producers to adopt it as a means of adding to the marketability of their products. In the context of ELRA, the term "validation" is used as a synonym to evaluation, that is, it refers to the activity of checking the suitability for the market, the adherence to standards, and the quality control of the product.

The ELRA validation work is applied to three areas of activity:

- **Speech** guidelines for validation procedures to be carried out in order to ascertain a certain quality standard of spoken language resources are distributed by ELRA. The methods proposed are chosen such that they are a good balance between achievable quality standards and associated costs of the validation procedure.
- **Text**: Aiming to fulfil its objectives regarding the production of a validation manual, ELRA works in close co-operation with highly recognised research centres in order to come up with validation manuals.

[1] www.icp.inpg.fr/ELRA

The work being carried out capitalises on previous projects including, but not limited to EAGLES.

- **Terminology**: methods and tools for validation and standardisation of terminological resources are being produced. These resources are essential to a variety of applications, such as translation, document management, and software localisation.[1] .

1.2.3 ISLE and Evaluation of MT Systems

ISLE (International Standards for Language Engineering)[2] is both the name of a project and the name of an entire set of co-ordinated activities regarding the Human Language Technology field. ISLE acts under the aegis of the EAGLES initiative and has produced a draft classification of machine translation evaluations. Its goals are:

- to work toward a theory about the methodology for evaluating Natural Language Processing / Computational Linguistics applications in general;
- to develop a general framework in which existing evaluation measures for particular language engineering applications can be formulated in a systematic and organized way;
- to illustrate the theory and methodology, and to take further previous work, by creating a specific framework for classifying evaluations of Machine Translation systems. This work involves gathering and classifying individual evaluation measures in the most suitable groupings, and creating criteria for the application of each measure.

While ISLE evaluation research is currently geared towards Machine Translation Evaluation, it is planned to be also tested with and adapted to other language engineering applications in the near future.

1.2.4 DiET and Glass Box Evaluation

DiET (Diagnostic and Evaluation tools for Natural Language Applications)[3] aimed to develop data, methods and tools for the glass-box evaluation of language engineering components, building on the results of previous projects covering different aspects of assessment and evaluation. It aimed to extend and develop test-suites with annotated test items for grammar, morphology and discourse, for the English, French and German languages.

---

[1] more information can be found at the University of Surrey website under http://www.surrey.ac.uk/ or under http://www2.echo.lu/langeng/en/le2/interval/interval.html of the European Commission

[2] http://www.ilc.pi.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm

[3] cf. http://www2.echo.lu/langeng/en/le3/diet/diet.html

1.2.5 ELSE Evaluation

The ELSE project (Evaluation in Language and Speech Engineering)[1] was contracted by the European Commission to study the possible implementation of comparative evaluation in Europe. It distinguishes between five types of evaluation:

- **Basic research evaluation** tries to validate a new idea or to assess the amount of improvement it brings over older methods.
- **Technology evaluation** tries to assess the performance and appropriateness of a technology for solving a problem that is well defined, simplified and abstracted.
- **Usage evaluation** tries to assess the usability of a technology for solving a real problem in the field. It involves the end-users in the environment intended for the deployment of the system under test.
- **Impact evaluation** is the evaluation of the socio-economic consequences of a technology.
- **Program evaluation** can be seen as an attempt to determine how worthwhile a funding program (like LE) has been for a given technology.

1.3 Conclusion

The mere fact that so many evaluation initiatives were called into life, and research work of so many scientists was directed towards the development and improvement of evaluation methodologies shows that the need for such an evaluation methodology is, in fact, striking.

Jargon florishes readily in evaluation research. However, despite all the differences in naming and focus, all approaches have in common, that they try to relate the human precept of how some system should behave to the actual performance of this system. The complexity of this problem has been addressed in each of the existing approaches and a great deal of effort has been put into the development of methods to solve these problems. Depending on the final goal of testing, the depth of understanding as well as the evaluation procedures may vary, while the central concepts remain the same. This is particularly true for the different types of evaluation as defined by EAGLES, that is, *adequacy*, *diagnostic* and *progress* evaluation. Though they are performed at different stages of the development cycle, and with a different focus, they share the central concepts.

[1] cf. http://www.limsi.fr/TLP/ELSE/

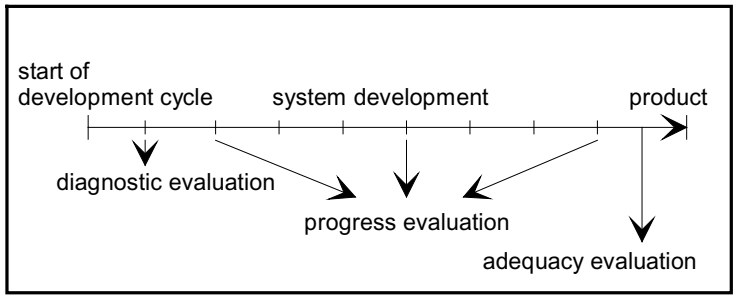


Figure 19: Relationship between EAGLES Evaluation Types

The framework for the evaluation of translators' aids' systems which will be presented in this thesis has evolved during more than a decade of evaluation work - both research and practical. Efforts of the above described initiatives and many practical evaluation examples have been taken into account, leading to a theoretically sound and practicable evaluation framework for translators' aids. While the framework shares the basic ideas with those developed in the context of the above initiatives, the major difference is that of practicability. The start-off of this framework was the practical need to produce evaluation results in the ESPRIT TWB Projects. Thus, many a theory was discarded at a very early stage due to the inability to produce any significant results. The evaluation procedures were applied with various translation tools developed by the different partners. When the TWB projects ended, the European Union considered the then evaluation procedures as one of the most outstanding research results produced in the TWB projects. At the time, the major achievements were the qualitative results gained by means of the test types developed for evaluation, and the approach used to improve and monitor the progress of the systems. At that stage, the framework had been geared towards evaluation supporting system development.

Being part of the EAGLES context, focus was shifted towards adequacy evaluation, that is, evaluation preceding purchase decisions in translation industry. Consequently A central issue was how to arrive at quantitatively measurable primitives that are needed for adequacy evaluation as opposed to implementable primitives needed in the system development context. When the EAGLES Initiative concluded, the base work for adequacy evaluation was founded, yet adequate means to formulate measurable primitives from user requirements were not yet found. Motivated by the tendency towards formalisation of evaluation procedures that was gaining ground during EAGLES, postgraduate research at the University of Helsinki led to a major breakthrough towards formalisation, quantification and assessment. Modelling procedures from Software Engineering were applied to the context of evaluation,

resulting in ways to produce measurable primitives from user models. Practical testing performed with the TRADOS Translators'Workbench and the IBM Translation Manager products could deliver quantifiable results. There was only one step left in the process that had to be dealt with, that is, assessment. This involved the process of relating the views of users back to the test results . The discipline of decision analysis, concerned with the motivation and rules behind multiple choice decisions provided instruments that could be adapted to the context of software evaluation and eventually led to the assessment of test results in terms of user needs.

2. What Translators Want - Featuring Users and Systems

According to the Dictionary of Contemporary English the term “want” denotes the condition of lacking something necessary or very useful. In the previous chapter the translation context was discussed and useful functions of a multilingual corpus tool were identified. Roughly comparing these functions to the functionality of the translators’ aids’ systems on the market, it became obvious that these systems roughly offer most functions that would be useful for translators. What could not be shown, however, was **the extent to which** the functionality of these systems actually corresponds with what translators want. The condition of wanting presupposes an awareness of two things, that is (i) needs, and (ii) possible solutions. In other words, in order to determine how useful a specific translators’ aids’ system is for a specific type of translator, the features of both translator and system have to be determined and mapped onto each other.

This type of mapping between the system on the one hand and the user context on the other is well known in the requirements engineering context. Jackson/Zave (1993:pp56) developed a model which describes the *nature* of requirements formulation. Their approach concentrates on the description of *domains* (or "real worlds") and requirements (or "problems") on the one hand and system properties on the other. In the place of the traditional term "environment" their approach makes use of the term "domain". They argue that "domain" is a broader concept that denotes the overall subject matter of the system's computations and provides the context in which those computations have useful meaning or effect. A domain is a topic for description in its own right, independently of any description that eventually will be made of the system to be constructed. Requirements are a special type of domain descriptions, which describe the desired state of affairs, while ordinary domain descriptions assert certain truths about the domain. Within the same domain different users may have different requirements, forming different subsets of the overall set of domain properties. The purpose of the system is to bring about observable effects in the domain. In formal terms the relevant space of descriptions in requirements formulation is described as covering two intersecting sets of attributes. The set $\{Di\}$ is a set of attributes of the *problem domain*, and the set $\{Mi\}$ is a set of *machine* attributes. The intersection of D and M is S, the area of specifications, where attributes exist in both the problem and the machine area. Figure 20 illustrates the intersection.

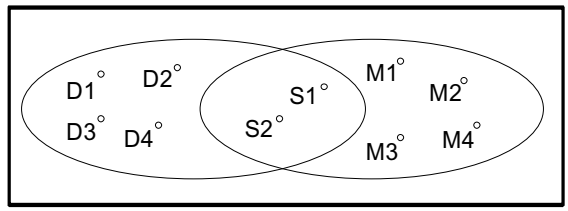


Figure 20: Domain and Machine Attributes and Specification Adapted from Jackson (1995:3)

In other words, the intersection {S} is a set of *implementable primitives*, that is, small units of functionality, both needed by the user and possible to be implemented. A detailed description of all elements of {S} provides the basis for the system specification.

In the same spirit a model can be developed for the evaluation of translators' aids' systems. However, while in the development context, the intersecting set denotes *implementable primitives*, in the evaluation context this set has to denote *measurable primitives*. Evaluation requires the description of the domain {D}, above all, the definition of tasks users perform (a subset of {D}) and features that programs offer to perform certain tasks (a subset of {M}). In evaluation the intersection of D ∩ M is E, that is, the evaluation space. The nature of the different sets in evaluation is as follows:

{*D*} a set of attributes stemming from the tasks users perform and the technical and organisational environment in which they are performed;

{*M*} a set of attributes, that is, features systems offer to perform specific tasks;

{*E*} the intersection set of attributes which are both relevant to users in the domain and covered by the system under evaluation.

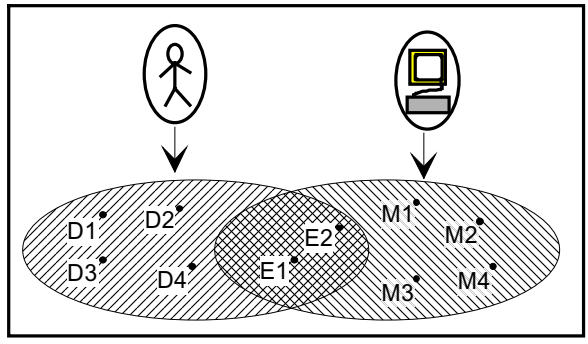


Figure 21: Model of Requirements Formulation for Evaluation

The meaning of the three sets in the context of evaluation may be characterised as follows:

> $\{D_{1...n}\}$ is a set of attributes of the problem domain representing different tasks, that are *not* covered by the system(s) under evaluation. $\{D_{1...n}\}$ may be different

for different users and systems. User-oriented evaluation has to determine the scope of $\{D_{1...n}\}$ and rate the importance of these attributes for users. If $\{D_{1...n}\}$ covers an attribute of overriding importance, the system under evaluation failed.

$\{M_{1...n}\}$ is a set of machine attributes that are *not* explicitly defined by users and as such are not directly relevant to evaluation. $\{M_{1...n}\}$ may be different for different users and systems.

$\{E_{1...n}\}$ is a set of evaluation attributes for which measurement ways have to be found, since they represent attributes of the problem domain which are provided by the systems under evaluation. $\{E_{1...n}\}$ may be different for different users and systems. Comparing the scope of $\{E_{1...n}\}$ for different systems leads to the identification of *evaluation relevant* attributes. In other words, only those attributes are relevant to evaluation in which the systems under evaluation differ. In decision analysis, Winterfeldt/Edwards (1986:41) point out that *evaluation relevant* attributes have to be considered in the light of their measurability, that is, only if they allow sensible measurement, attributes are *value relevant*. Testing should deliver attribute/value pairs for the different systems with respect to all attributes of $\{E_{1...n}\}$ that are both evaluation *and* value relevant.

2.1 Elicitation of Attributes of {D} – the Needs of Translators

As the evaluation model shows, the first step in defining measurable primitives is to look at the domain {D} and the machine {M} in their own right. Jackson/Zave's (1993:pp56) distinction between ordinary domain descriptions and requirements, which describe the desired state of affairs, is also relevant to evaluation. The aim of this section is to define domain properties, that is, to assert certain truths about the translation domain. The definition of requirements which describe the desired state of affairs is topic of chapter 3.

The process of eliciting domain properties for evaluation can benefit from the experiences made in requirements analysis and elicitation in the software development context. Sommerville (1996:pp88) notes that requirements elicitation and analysis is generally difficult because stakeholders, that is, everyone who may have some direct or indirect influence on the system requirements, often do not really know what they want or can expect from computers, except in the most general terms, and

consequently, make unrealistic demands. Sommerville/Sawyer (1997:63-110) present thirteen guidelines for eliciting requirements in the software development context.

REQUIREMENT ELICITATION GUIDELINE	KEY BENEFIT	COST introduction	COST application	beneficiary
1. Assessment of system feasibility	Reveals if a system is actually needed and technologically realistic.	Low	Low to moderate	SE/ users
2. Sensitivity to organisational and political considerations	Helps the software engineer understand why some requirements are suggested	Low	Very low	SE/ users
3. Identification and consultation of system stakeholders	Discovery of all likely sources of requirements	Very low	Low	SE
4. Recording requirements sources	Requirements traceability from original sources	Low	Low	SE
5. Defining the system's operating environment	Fewer installation problems for delivered system	Low	Low	SE
6. Using business concerns to drive requirements	Requirements are focussed on core business needs	Low but senior managers to be included	Low	Users/ SE
7. Investigating domain constraints	Domain constraints lead to the identification of critical requirements	Low	Moderate	SE
8. Recording the rationale for requirements	Improves the understanding of requirements	Low	Low – Moderate	SE
9. Collecting requirements from multiple viewpoints	Better requirements coverage	Moderate– High	Moderate	SE
10. Prototyping (poorly understood) requirements	Better understanding of the real needs of system **users**	Moderate	Low – high	Users
11. Using scenarios to elicit requirements	Users find it easy to understand scenarios and to describe associated requirements	Fairly high	Low	Users
12. Defining operational processes	Reveals focussed requirements and requirements constraints	Fairly high	Moderate	SE
13. Revising requirements	Lower cost, faster elicitation of requirements	Moderate – High	Moderate	SE/users

Figure 22: Overview of Key Guidelines for Requirements Engineering According to Sommerville/Sawyer (1997)

In the context of the elicitation of domain properties for evaluation, major focus must lie on

- guideline 2, that is, a sensitivity to organisational and political considerations;
- guideline 3, that is, identification and consultation of system stakeholders;
- guideline 6, that is, using business concerns to drive requirements;
- guideline 7, that is, investigating domain constraints;

- guideline 9, that is, collecting requirements from multiple viewpoints; and
- guideline 12, that is, defining operational processes.

Applying these guidelines to the context of eliciting domain properties for the evaluation of translators' aids, the following dependencies in the translation domain {D} can be identified:

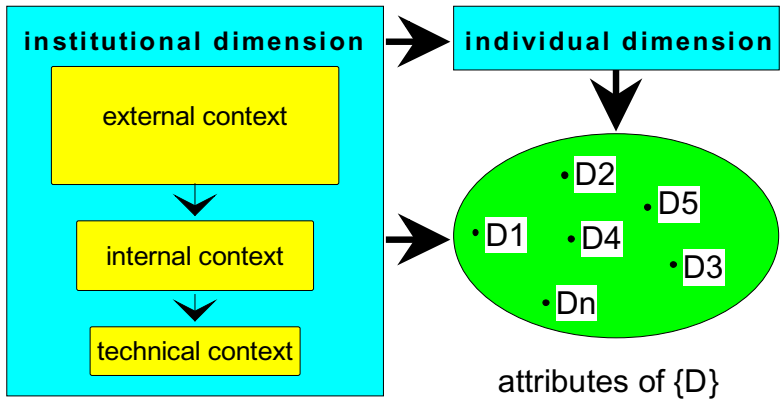


Figure 23: Configuration of {D} from Institutional and Individual Dimension

The institutional dimension denotes the organisational environment and covers those attributes of {D} that are *independent* of the individual. As the above figure illustrates, it is useful to distinguish between the external context, the internal context and the technical context.

The elicitation of domain properties for translators' aids can be performed along a pre-established list of parameters that are relevant in the translation domain. The lists which can be found below are based on and further develop the user requirements studies performed by Fulford/Höge/Ahmad (1990) in the context of the ESPRIT TWB project No. 2315, and the requirements analysis part developed in the context of EAGLES (1995). They should be used as a *guideline* that outlines the most important parameters, while each specific environment may still ask for the adaptation or addition of parameters.

The *external context* in translation relates to those aspects that serve as environment to the translation activity.

EXTERNAL CONTEXT	
PARAMETER	POSSIBLE CATEGORIES
nature of enveloping organisation	agriculture; industry; services... domestic company; international company ...
size of organisation	employees; turnover...
international activity	amount; nature...
language policy	national language as official language; foreign language in common use; foreign language as official language; national and foreign language as official languages; several foreign languages in common use; several languages as official languages ...
documentation policy	quantity of text produced in foreign language; languages in which documentation is distributed; annual translation volume per language; ...
terminology policy	joint resources with other companies; company resources; department resources; individual resources; ...

Figure 24: Parameters of the External Context

The *internal context* of translation relates to details of the translation work.

INTERNAL CONTEXT	
PARAMETER	POSSIBLE CATEGORIES
type of translation organisation	freelance; translation company; centralised translation activity; decentralised translation activity; subcontracted translation activity; bi/multilingual organisation...
texts	source language(s) target language(s) text types domain (or subject area) characteristics frequency of translation ...
quantity of translation work	pages/time
quality of translation work	raw translation; normal quality; high quality ...
extent of translation work	proof-reading of foreign language texts; updating existing translations; carrying out new translations; editing and translating text; editing, translating, establishing text layout; interpreting; ...
extent of terminology work	using existing terminology; updating terminology; elaborating new terminology; checking terminology; ...
type of job	professional translator; interpreter; editor; domain expert; support personnel;...
role of personnel	typist; translator/interpreter; terminologist; proof-reader; head of language groups ...

Figure 25: Parameters of the Internal Context

The *technical context* of translation concerns technical details in which translation is embedded.

TECHNICAL CONTEXT	
PARAMETER	POSSIBLE CATEGORIES
hardware environment	type of computer; power; storage capacity; ...
storage media	hard disk; diskette; tape; CD ROM; ...
operating environment	DOS; OS/2; UNIX; Windows; ...
software environment	desktop; text processing; databases...
documentation management	hypertext; imaging; full text retrieval; indexing; ...
nature of documents	text; graphics; photographs; animation; video; ...
text input form	dictation; hand-written; typewritten; printout; floppy; network; e-mail; OCR; ...
text output form	dictation; hand-written; typewritten; printout; floppy; network; e-mail; ...
text transmission	post; FAX; network; e-mail; ...

Figure 26: Parameters of the Technical Context

Constraints that are set in the *external context* of translation largely determine the *internal context*, which, again, determines the *technical context*. If, for instance, the nature of the enveloping organisation (external context) is car manufacturing, the characteristics of the texts (internal context) will include graphics, which require a software environment (technical context) that can deal with graphics.

The *individual dimension*, finally, covers those aspects that are *dependent* on the individual, that is, which may be different from one user or stakeholder to another even within the same institutional setup.

INDIVIDUAL DIMENSION	
PARAMETER	POSSIBLE CATEGORIES
knowledge/experience	native language(s) foreign language(s) language direction(s) language proficiency domain proficiency years of experience ...
tasks	type of tasks sub-tasks (manual/computer supported) frequency of task rating of importance of task language directions typical problems problem solving strategies reference materials type of information from reference materials type of support ...
terminology terminology	type of terminology support type of terminological information strategy for terminology elaboration aids for terminology elaboration terminology maintenance integration of terminology work frequency of terminology work ...

Figure 27: Parameters of the Individual Context

Among the most important individual parameters are the individual's knowledge and experience both in the working domain and with computers. As pointed out before, experience and knowledge are responsible for the development of individually different strategies for problem solving, and consequently, to some extent, determine which characteristics of a translation support system are considered important. There is, however, a strong dependence between the individual and the institutional dimension: constraints which are put on the individual by his/her institutional setup and the role which the individual plays within the larger context of the institution, are also responsible for the type of tasks and problem solving strategies applied by the individual. For instance, one and the same translator will have different requirements in terms of translation support systems, if working as freelancer or within a centralised translation department. People who work in the same context are likely to perform similar tasks. If tasks are similar there may be an overlap of attributes relevant to evaluation, despite individual differences.

Knowing which information has to be gathered during the elicitation process, however, is only part of the whole process. The next question is, where to get the information from, that is, who is the ideal information provider. The distinction between the *institutional* and *individual* dimension of translation is particularly useful

when identifying appropriate information providers. Accordingly, data on the *external context* of translation has to be elicited from persons in the management board; data on the *internal context* from both management and translators; data on the *technical context* of translation from technical support personnel; and, finally, data related to the *individual dimension* from translators.

Apart from determining the type of information that needs to be elicited and the information provider, the elicitation of domain properties asks for the choice of the adequate elicitation method. The *institutional dimension* denotes the general background to the translation activity. It establishes a picture of *facts* which determine the environment of the translation process. According to Wilson (1990:200) the most appropriate elicitation techniques for the elicitation of *facts* are questionnaires and interviews.

Oppermann (1988:10) points out that questionnaires are frequently used for all phases of software development and evaluation. They are used to elicit both quantitative and qualitative data. Goguen/Linde (1993:156) argue that they can be useful instruments when the population is large enough and the issues addressed are clear enough to all concerned. The reliability of results arrived at by means of questionnaires strongly depends on the number and representativity of persons questioned. Oppermann (1988:10) complains that questionnaires are likely to only deliver those results that are welcome to the designers of the questionnaire, that is, the choice of questions biases the results. This is due to the fact that the way of posing questions may implicitly suggest the "correct" answer.

There are various possibilities to perform interviews. Diaper (1989-3:229) and other knowledge engineers distinguish between focussed and structured interviews. In a focussed interview, the interviewee is prompted with a question related to his/her working environment, that is, typical tasks, problems etc. and his/her general opinion towards the system under testing. The interviewee is thereafter given the opportunity to express him/herself freely while being interrupted as little as possible. The principal aim of the focussed interview is to obtain a typology of objects and agents in the domain, to establish basic factual knowledge, and to achieve a breakdown of the problem. The structured interview is used for obtaining detailed information on specific topics. Goguen/Linde (1993:154) point out that the success of the interview turns on the premise that (a) relevant questions can be decided in advance of the interaction and (b) questions can be phrased in such a way that, as long as they are read without variation, they will be heard in the intended way and will stimulate a

valid response. However, making use of natural language, the interview is inherently available for multiple interpretations of the meaning of both questions and answers.

The *individual dimension* is concerned with details of the translation activity. Elicitation should lead to the recognition of *strategies*, *causal knowledge*, *procedures* and *rules*. The most appropriate elicitation techniques for this type of knowledge are interviews and observations. On page 148 Cordingley (1989) describes observations as the activity of noting and recording features of 'naturally occurring' settings, and of the events and actions within them, either directly or indirectly by means of video or one-way mirrors. During observation one or more observer(s) sit close to the subject, while watching and taking notes. Diaper (1989-3:213) complains that observation is a very delicate matter indeed, since it is extremely difficult to record sequences of behaviours in their correct order. Though video recording is not as intruding as direct observation, its effect on the behaviour of the user should not be underestimated. There is an ongoing debate as to the extent to which people alter their behaviour, once they become aware that they are being observed. According to Cordingley (1989), there is a body of opinion that there is no significant alteration; another that although people may adapt their behaviour initially, they soon forget the observer and revert to their usual behaviour patterns; another that although there may be alteration it does not invalidate the material; yet another that it is possible to take the alterations into account while interpreting the data; and finally some suggest that the technique is so flawed as to be at best useless and at worst misleading.

Cordingley (1989:pp.170) presents detailed so-called *Personalised Task Elicitation Questions* which are a good starting point for either interview or observation leading to the elicitation of information about tasks that may be relevant in evaluation. They cover the following topics:

(1) Basic description
What: what is done

(2) Temporal ordering
Before: what processes come before it in time and have a message or a material flow leading to it
Next: what processes come after it in time and have a message or a material flow leading from it
Concurrent: what processes happen to occur at the same time but which do not share a common 'before' or 'next' relationship to it

(3) Contingency information

Or: alternative processes; which one is done depends on predetermined control conditions ('Or' processes do dot send messages or materials to one another)

And: all processes are to be done but in any order ('And' processes may or man not send messages to one another)

(4) Establishing hierarchies

Why: one is done for the purpose of the other(s); 'Why' relationships establish superordinate/subordinate relationships in hierarchies of purposes; usually the superior sends a control message to the subordinate and receives a data message (a report on progress) back from it

How: one is done as a means of achieving the other(s); 'How' relationships establish superordinate/subordinate relationships in enabling hierarchies

(5) Production information

Control: control messages start and stop processes; the express conditions for activating processes; identify their source(s) and destination(s)

Concurrent controls: all messages have to be present and all have to arrive at the same time for the process to be activated

'or' controls: if any of the messages is present then the process is activated

'and' controls: all messages have to be present but they can have arrived in any order for the process to be activated

Data: messages which are the informational inputs to processes; identify their source(s) and destination(s) and whether they come and go directly or via a store (a 'pool')

Materials: the physical inputs and outputs of processes

Products: the outputs of the process; they may be messages (data or control) or materials

Tools: what is used by people to help them carry out a process; distinguish between types in terms of the process the tool is aiding

(6) Scope information

Boundaries: Define in terms of the start (successive before?), the end (successive next?), the top level purpose (successive why?), and functional primitives (successive how?)

Who: the agent, object or processor doing the process

Where: the physical location of the process, message or material flow

Linked to: non-functional relationships such as 'similar to'

(7) Evaluative information

How well: attainment compared against some goal
How liked: how (the full range of) target users like doing it
How easy: whether (the full range of) target users find it easy to do

(8) Ergonomic information

Health, safety comfort: identify 'hazards'

To conclude, the aim of section 2.1 was

(i) to provide a brief insight into the complexity of requirements elicitation from the software engineering point of view;
(ii) to produce an exhaustive list of parameters and questions that might be relevant to the elicitation of properties in the translation domain {D}; and
(iii) to describe some basic aspects of the elicitation techniques that are appropriate for the type of information elicited.

The effort that is put into needs elicitation has to depend on the size of the evaluation project, that is, whether performing evaluation on behalf of large organisations like the European Union, producer organisations and the like, or whether to perform evaluation on behalf of some translation department or agency. While in the former case large questionnaire surveys, observation and interviewing actions will be necessary on a representative number of translators, in the latter case, the interviewing of several translators, eliciting their tasks and background, will be sufficient.

The following task description is one possible outcome of the elicitation process which leads to the definition of the truths about a domain. The task description is based on the experiences gained by the author while doing research at the Mercedes-Benz translation department between 1989 and 1994. It distinguishes between administrative, technical, preparatory and operative tasks related to the overall process of translation in a computerised translation environment.

TASKS	SUBTASKS	ACTIONS
administrative	project organisation	assigning project codes defining text attributes distributing source texts among translators calculation of prices monitoring of deadlines invoicing
technical support	installation	installing programs adapting environment
	configuration	configuring editor configuring termbank configuring translation memories
translation preparation	terminology preparation	extracting new terms from texts elaborating terminological information updating terminology importing terminology exporting terminology producing terminology lists/dictionary printouts
	translation memory preparation	configuration of new TM databases updating databases alignment of parallel texts importing databases exporting databases
operative translation tasks	SL text reception	starting programs opening SL text creating TL document
	terminology	starting of termbank opening of termbank(s) accessing termbank from editor searching terms browsing in termbank selecting termbank entries pasting terms into text editing terminology updating termbank entering new terms
	translation	starting translation memory opening translation memory/ies creating new translation memory selecting attributes for translation memory changing fuzzy match percentage retrieving sentences from translation memory choosing translations from matches retrieving parts of sentences/terms from translation memory updating translation memories editing translations in translation memory databases entering/saving new translations into translation memory
	TL text delivery	spell checking saving new translations printing copying to floppy/network drives mailing TL texts exiting programs

Figure 28: Example for Task Description as Outcome of Featurisation of {D}

The above task description shows what kinds of actions a system that would be useful for the above context must support in one way or the other. It can be used as the starting point for the definition of measurable primitives relevant to this specific environment which will demonstrated in chapter 3.

In general the elicitation of properties of the translation domain along the above presented parameters and questions, making use of the discussed elicitation techniques, will deliver a broad variety of domain properties of different translators. Determining regular variations of these properties, it is possible to define so-called user profiles, which, according to Douglas (1995:4) behave like a parameterisation of domain properties and requirements statements.

In the software development context, user profiles can be used to distinguish between different types of interaction with the system (e.g. in Windows login, that of administrator, vs. user). Mayer (1993:93) developed different user interfaces for the interaction with a terminology database, depending on user profiles (translators and terminologists). When consulting the terminological database, each user can set specific parameters that are relevant to his/her interaction with the terminological database and the user interface is adjusted accordingly, for instance, in terms of dialogue language or information categories of the termbank.

In the context of evaluation, user profiles could be used to determine typical needs of specific types of translators and contexts. Depending on the typical tasks translators or users of translators' aids' systems perform, similar attributes and measurable primitives are likely to be relevant. The above task description could be considered a starting point for the definition of a user profile for translators in mid-to-large size translation departments. Future research should be directed towards this area, particularly in view of the reusability of resources. If it is possible to establish detailed lists of measurable primitives for specific types of users, these lists should be made generally available. The future evaluator could then simply select the user profile that comes closest to the context of the evaluation scenario, and adapt it to the specific circumstances, thus saving a great deal of time and effort.

2.2 Elicitation of Attributes of {M} – the Functionality of Translators' Aids' Systems

The principal motivation behind the elaboration of {M} in user-oriented evaluation is to allow the identification of possible elements of {E}. In other words, it has to be determined whether there is an overlap between the tasks that are described in {D} and the functions of {M}. Therefore, for evaluation purposes, the elaboration of {M} is

driven by the nature of the tasks that are central to {D}. Once it is clear, which tasks are central to evaluation, the first step is to examine whether the systems under evaluation offer functions to perform the given tasks. This can be achieved by studying the system documentation or specification. This process leads to the identification of {E}, that is, those attributes that have to be considered during the evaluation process. It moreover leads to the identification of features of {D} that are **not** covered by {M}. If these features have a high priority in the domain, they may function as knock-out criteria and, therefore, lead to the termination of the evaluation process for this specific system.

The elicitation of features of {M} differs with respect to the two evaluation situations. When evaluating the adequacy of *existing* systems for translation industry, the features of the different systems are a *given*. Considering central functions and their features as they are described in the system documentation and comparing it with the central tasks as elicited before, helps to answer the following questions, which are central to any evaluation process (Winterfeldt/Edwards, 1986:41):

- Are there *dominated* systems, that is, systems that do not offer functions to perform central tasks that other systems offer? If yes, the dominated system can be dropped.
- Which are the *evaluation relevant* attributes, that is, where is it likely that the systems differ in terms of their performance? Only evaluation relevant attributes should be considered for later evaluation.

When performing evaluation in the context of translation system development, the attributes of {D} should contribute to the definition of {M}. Ideally this step of the evaluation process starts *before* system specification and development. If the system specification is already finished, the features of {D} should be compared to the features listed in the system specification. Important questions to be tackled at this stage are:

- Is the implementation of all features of {D} planned?
- If not, what are the reasons for not considering specific features of {D}? Are there technical restrictions to the implementation of a specific feature of {D}? Is it a question of time/effort?
- How important are those features of {D} that are not implemented to the user?

Further evaluation processes depend on the nature of {E}, that is, the overlap between {D} and {M}. The following figure is an example of how the mapping between {D} and {M} can be performed in both evaluation situations for the task of *terminology preparation*. If evaluation precedes a purchase decision the table allows the identification of *dominated* systems. If evaluation supports development, the table

considers at this stage, whether functions to perform the subtasks are part of the system specification {S}, and roughly how things are planned to be implemented or, if not, what the problems are.

SUBTASK	EVALUATION PRECEDING PURCHASE function available?			EVALUATION SUPPORTING DEVELOPMENT	
	SYS 1	SYS 2	SYS 3	in {S}?	COMMENT
extracting new terms from text	0	0	0	0	problem: identification of terms vs. words
elaborating terminological information of new terms	0	1	0	1	term can be located in text corpus, context, grammar info elaborated semi-automatically
updating terminology	1	1	1	1	different interfaces for different user types planned
importing terminology	1	0	1	1	only TIF format
exporting terminology	1	1	0	1	as on-line termbank and dictionary type printouts
producing terminology lists	0	1	1	1	along different filters

Figure 29: Mapping {D} and {M} for Terminology Preparation Task

The above table shows that there are *no* dominated systems, since there is no system that is worse than others in all respects. When supporting system development it makes sure that every technically possible function is considered.

To conclude, the chapter "What Translators Want" elaborated how the basic properties of the translation domain can be determined and how they can be mapped onto the features of translators' aids' systems, resulting in a set of features of {E}. The next step in the process of evaluation, which will be described in chapter 3, is that of defining translator's requirements, that is, describing the desired state of affairs for each of the features of {E}. This description asks for a quality definition of each evaluation relevant feature, and the development of a scale on which each feature can be measured.

3. Structuring and Preparing for Evaluation

In the previous chapter it was discussed, how truths about the translation domain can be elicited and, roughly, how they can be mapped onto the functions offered by translators' aids' systems. The next step, that is, the development of measurable primitives for evaluation can be considered a top-down approach in which the domain presents the top node. The structuring of this domain leads to a great number of measurable primitives for which values have to be obtained. Consequently ways of structuring and measuring the translation domain have to be investigated. The structuring of evaluation problems consists of defining and organising the objectives, attributes and values on which different alternatives under evaluation should be compared. There is ample evidence that in the context of the evaluation of social science programmes, decision analysis procedures have been successfully applied since the 1960ties (cf. Struening/Guttentag (1975); Guttentag/Struening (1975), Edwards/Newman (1982)). In the context of software engineering, the structuring of domain problems is reflected in the procedures of quality requirements definition and modelling. In section 3.1 basic approaches from decision analysis (3.1.1) and software engineering (3.1.2) will be presented and applied to the problem of evaluating translators' aids' systems. In section 3.1.3 the impact of the different approaches on a framework for the evaluation of translators' aids' system will be discussed.

In section 3.2 methods for preparing the evaluation process will be presented for the two evaluation contexts that are relevant to this framework, that is, evaluation preceding purchase decisions (3.2.1), and evaluation supporting software development (3.2.2).

3.1 Basic Approaches

Evaluating the adequacy of different options for a specific context implies that the evaluator has to make decisions at various levels. How these decisions are made on a rational basis has, so far, been neglected in the context of software evaluation research, which has primarily been based on findings from software engineering. In order to decide whether and to which degree it makes sense to apply decision analysis principles to the evaluation of software systems, the different approaches of the two disciplines have to be considered in more detail.

3.1.1 Approaches from Decision Analysis

Decision analysis is concerned with the evaluation of alternatives with respect to multiple attributes. Recent text books and monographs on decision analysis include Meyer (1999), Byrnes (1998), Golub (1997) or Schick (1997); For recent learned articles in this area see Rapoport ed. (1998); and Bouyssou ed. (1998). There is an overlap of concepts between decision theory and many other disciplines such as measurement theory, behavioural research, economics, statistics, psychology, philosophy, and, sociology. For the evaluation of translators' aids' systems the overlap between decision analysis and measurement theory is of particular interest: describing quantitative representations of qualitative relations, and delineating the circumstances in which such representations are possible and in which they are not. Keeney/Raiffa (1993) is a key text for this area. Central concepts are: structuring problems; representing these in quantitative terms; performing utility measurement for multiattribute problems; and performing tradeoffs.

Apart from measurement theory, the overlap between decision analysis and behavioural research is particularly relevant to the user-oriented evaluation of translators' aids' systems, since it involves the description, structuring, and measurement of qualitative information elicited from people. Winterfeldt/ Edwards (1986) and Edwards/Newman (1982) are key texts in this area, discussing among other things problems related to (i) eliciting judgments from people; (ii) handling subjective elements in value structuring; (iii) performing experimental validation; (iv) handling consistency and reliability issues of value and utility judgements.

3.1.1.1 The Structuring Problem

The task of specifying what is relevant to a specific problem domain, of structuring what the options are, and of defining the relation between options and outcomes is what decision analysts call the *structuring problem*. The intertwined processes of articulating objectives and identifying attributes are basically creative in nature and cannot be performed sequentially, step-by step. According to Winterfeldt/Edwards (1986:pp.29) when elaborating details of the problem, the analyst must understand two things: the vocabulary of the problem area and the structure of the organisation. The primary methods employed in decision analysis to gain an insight into both vocabulary and organisation structure is to examine written material and organisational charts and to interview employees. Edwards/Newman (1982:pp.33) point out that for each problem the stakeholders have to be identified. A stakeholder is a person that has an interest or a stake in the object being evaluated. Individual stakeholders are likely to have different views on the problem and its environment. The values of the different

stakeholders have to be examined and their influence on the decision process determined.

The principal assumption in decision analysis is that if you know what your problem is, what your options are, and what values bear on their merits, you can make assessments and then do arithmetic that will lead you to an instrumentally rational act. It is possible to structure problems in form of *value trees* which delineate the decision maker's preferences. The term *value tree*, which will be used in this thesis, goes back to Winterfeldt/Edwards (1986), other decision analysts such as Keeney/Raiffa (1993), Keeney (1980), Hogarth (1985); use the term *objectives hierarchy*, while French (1989) uses the term *hierarchy of attributes*. Value trees are hierarchies of objectives and attributes on the basis of which different alternatives can be evaluated. They are "and" trees which map the decision maker's problems onto the structure of the objects under evaluation. According to Winterfeld/Edwards (1986:36) central questions when developing value trees are:

- what are the major objectives and concerns of the decision maker?
- what attributes differentiate the different alternatives under evaluation?
- how can these attributes be measured?
- how are attributes, objectives and values related?

Keeney/Raiffa (1993:pp.38) discuss the characteristics of attributes of a value tree. In general, characteristics of value trees should be *comprehensive*. An attribute is defined as *comprehensive* if, by knowing the level of an attribute in a particular situation, the decision maker has a clear understanding of the extent that the associated objective is achieved. Some attributes are *objective*, that is, there is a commonly understood scale for that attribute and its levels are quantitatively measurable (e.g. response time measured in seconds). When there is no objective index available (e.g. in the case of like or dislike of particular functions), *subjective* scales must be constructed. It is important to note that, though striving for quantifiable values, many decision analysts admit that various attributes are not objectively measurable. Winterfeldt/Edwards (1986:41) point out that the direct judgement of such attributes is a measurement procedure like any other. Faithful representation of an inherently subjective value structure, not objectivity, is the goal of structuring value trees. An attribute should be *measurable*, that is, if possible levels of the attribute can be assigned, and the decision maker's preferences can be assessed in terms of utility functions or rank ordering. In decision analysis, measurable attributes are frequently called *value relevant.* Furthermore*,* attributes that are taken up in the final decision process should be *evaluation relevant,* that is, if all alternatives under evaluation score the same on a given attribute, that attribute is not relevant to evaluation and should not be part of the

final calculation process. An attribute is *judgmentally dependent*, if the evaluation of an alternative with respect to one attribute depends on how the alternative performs with respect to the other. It is *environmentally correlated*, if the value of one attribute strongly influences the value of another. Winterfeldt/Edwards (1986:44) illustrate this type of dependency by means of the following example: if the decision has to do with production volume, for a manufacturing plant, cost of production and cost of distribution are environmentally correlated, since it costs more to ship more units.

In principle one should strive for straightforward assessment, avoiding problems like *judgmental dependency* or *environmental correlation*. Often difficulties can be removed by restructuring the parts of the tree that produce the problem. For instance, the problem of *environmentally correlated* attributes can be dealt with by combining the two correlated attributes into a new one. Winterfeldt/Edwards (1986:44) give the following example: in an apartment evaluation problem, the two *environmentally correlated* attributes *distance from campus* and *facilities for transportation* can be combined into *accessibility of the campus*.

When constructing value trees, the analyst should stop disaggregating when the dimensions at the lowest level are measurable, and easy to assess judgmentally. Figure 30 is an example of a value tree presented by Winterfeldt/Edwards for evaluating energy technology.

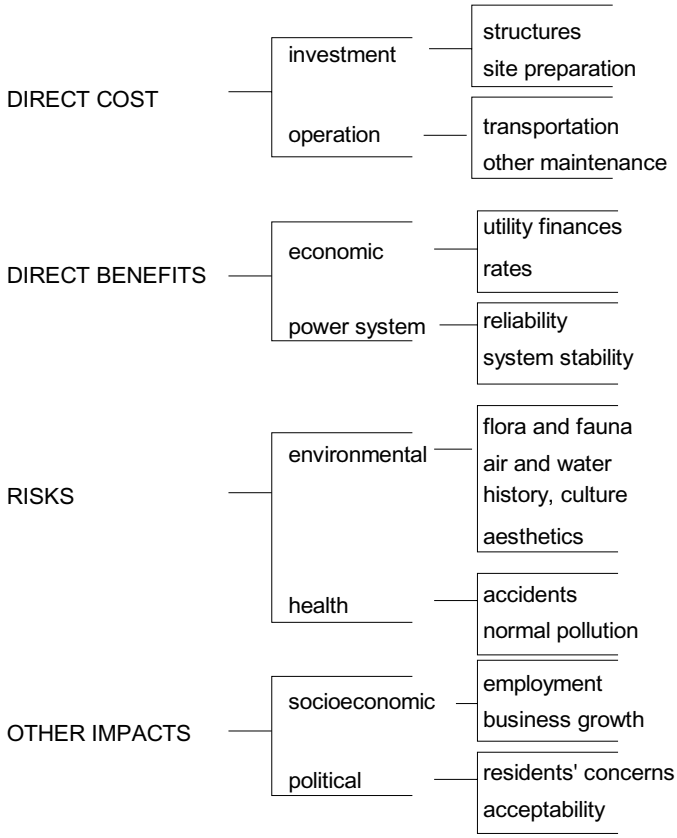


Figure 30: Value Tree for Evaluating Energy Technology by Winterfeldt/Edwards (1986:49)

Keeney/Raiffa (1993:43) argue that when dividing an objective into subobjectives on the one hand, care must be taken to ensure that all important issues of the higher objective are accounted for in one of the subobjectives. On the other hand, a proliferation of the tree may be both not necessary for the decision procedure and unmanageable. The "test of importance" is used to filter out those attributes that are not relevant to the particular decision process. The test of importance implies that before any objective is included in the hierarchy, the decision maker is asked whether he feels the best course of action could be altered if that objective were excluded. An affirmative response would imply that the objective should be included, a negative response that it should not. Winterfeldt/Edwards (1986:41-45) argue that the relationship between the lower level dimensions and a higher dimension should be hierarchical and directed; it should avoid cross-links with other higher level value categories, and create an exhaustive and nonredundant list of explanatory value dimensions. Keeney/Raiffa (1993:51-53) and Winterfeldt/Edwards (1986:43) consider the following, often conflicting, criteria relevant to examine the validity of a value tree:

- *completeness*: all relevant values are included;

- *operationality*: the lowest level values or attributes are meaningful and assessable;
- *decomposability*: the attributes can be broken down into parts and are judgmentally independent;
- *absence of redundancy*: no two attributes or values mean the same thing;
- *minimum size*: the number of attributes should be kept small enough to manage.

The quality of a tree may be clear only after assessment of the numerical values. Generally, attributes are easiest to think about if either more is preferred to less, or less is preferred to more. However, sometimes attributes may be nonmonotone, that is, have an ideal point or saturation level which will be discussed later.

According to Winterfeldt/Edwards (1986:pp.44) the use of *value trees* for evaluating options is straightforward: the analyst obtains values for the operational attributes (branches of the tree), converts the values to utilities, weights the attributes and carries out the appropriate calculations to generate an overall evaluation of the option. In practice, however, this approach has difficulties. For instance, there may be too many branches to carry out a sensible evaluation, or some branches of the tree may be irrelevant because the options do not differ in their performance on them. According to Keeney/Raiffa (1993:45) the vertical depth of the proliferation of the hierarchy does not necessarily force the analyst to quantify preferences down to this level of detail. The hierarchy after a given level may serve merely as a qualitative checklist that helps to think more clearly about higher level attributes. Simplicity and the ease of the judgmental task must be balanced against the operationality of the attributes.

In the context of the evaluation of translators' aids' systems it will have to be examined whether value trees can be mapped onto software quality characteristics, since they both represent a hierarchy of attributes relevant to the domain. The different characteristics and problems related to developing value trees may also be relevant when developing metrics measuring the quality of translators' aids' systems.

3.1.1.2 Scale Construction

When the attributes relevant to evaluation are identified, it has to be determined, how an attribute can be quantitatively represented. Measurement theory forms the basis for any type of numerical representation used in evaluation and decision processes. The following scale construction issues which are relevant to evaluation follow, above all, Winterfeldt/Edwards (1986); Keeney/Raiffa (1993); French (1986), Baird/Noma (1978); Nunnally (1975); Durham/Durham (1975); Wilson/Wilson (1975).

Attributes of {E} can be represented by different types of values , that is, numerical, binary, and nominal:

A *numerical* value includes any number. Typical examples for metrics that deliver numerical values are the measurement of the size of objects such as programs in byte, or the number of databases entries, the measurement of processing time etc.

A *binary* value can only take two possible forms, that is, 0 or 1. Metrics that typically take binary values are those measuring the availability of a feature, where 0 means 'not available' and 1 'available', or also those assessing statements in boolean terms where 0 means 'false' and 1 means 'true'.

A *nominal* value represents a qualitative characteristic of a system. A typical example for a metric delivering a nominal value in the language engineering context is: languages treated by parser? English, German,....

Qualitative relations and numerical representations are related in form of scales, in which relations between different attributes (empirical relative) can be mirrored by analogous relations between values on these scales (numerical relative). In other words, a scale is a function on the objects in the system which provides the numerical representation of the qualitative relations. Scales relevant to evaluation purposes are *nominal*, *ordinal*, *rating* and *ratio* scales.

A *nominal scale* establishes the relationship of nominal values between different systems. There is neither order nor equidistance within the possible set of answers. A typical example for a nominal scale in the language engineering area would be the metric: *operating system*.

systems under testing	operating system
sys 1	DOS
sys 2	Windows 3.11
sys 3	Windows 95
sys 4	OS/2
sys 5	Windows 3.11

Figure 31: Example for Nominal Scale

In general, n-valued nominal scales can be transformed to binary nominal scales by the mapping $A \rightarrow 2^A$, or, in other words, a nominal value can be measured by either its existence (1) or its non-existence (0). If, for instance, the nominal value of the metric *operating system* is 'Windows 95', the binary nominal scale would classify a system with **1**, if it works under Windows 95 and with **0** if it doesn't. Binary nominal scales are used in checklists.

The binary nominal scale also allows the combination of judgmentally dependent nominal values. If, for instance, attribute **A** 'Windows 95' asks for the existence of attribute **B** 'pentium processor', the binary nominal scale allows the representation of this problem in the form:

SYSTEMS	1	2	3	4
Attribute A	1	1	0	0
Attribute B	1	0	1	0
A ∪ B	1	0	0	0

Figure 32: Binary Scale for Combination of Dependent Attributes

This type of combination of features can be performed on n number of pairs of judgmentally dependent attributes, always delivering a binary value as result.

Given that each nominal attribute in a list of attributes is judgmentally independent and has the same weight, the extent to which different systems fulfil a set of attributes can be measured by summing up the binary values for the attributes. The system which scores highest is the best choice. The following scale illustrates this:

	System 1	System 2	System 3
Attribute 1	0	1	0
Attribute 2	1	1	0
Attribute 3	1	0	1
Attribute 4	0	1	1
Sum	2	3	2

Figure 33: Combination of Nominal Attributes on Binary Scale

As the above table shows, given independence and same weight, system 2 would be superior to system 1 and 3, since the sum of binary values representing the existence /non-existence of attributes is higher than for 1 and 3.

An ordinal scale presents the relation of values that are members of a pre-defined ordered set, where value 1 < value 2 < value n. There is no indication of "how much" in an absolute sense any of the objects possess the attribute, neither is there any indication of how far apart the objects are with respect to the attribute. A typical example for a metric that can be represented in an ordinal scale is *understandability of definitions delivered with termbanks*, where the set of values would be {not understandable < mostly understandable < understandable} or in numerical terms: {0, 1, 2).

systems under testing	understandability	numerical values
sys 1	understandable	2
sys 2	not understandable	0
sys 3	mostly understandable	1
sys 4	understandable	2
sys 5	mostly understandable	1

Figure 34: Example for Ordinal Scale

For a comparison of ordinal values the principle of dominance has to be considered, where

a dominates **b** if

$a_i \geq b_i$ for i = 1,2,...q with $a_i > b_i$ in at least one case.

The efficient set also known as the undominated set or the Pareto optimal set (Keeney/Raiffa (1993:70) is the set of undominated alternatives:

efficient set = {**a** ∈ A| there does not exist **b**∈ A such that b dominates a}

The importance of the efficient set is that the decision maker can confine his attention to it, discarding all other alternatives, because a dominated alternative can never be optimal.

A given type of task solution may require that certain minimum values are attained, which Keeney/Raiffa (1993:78) call boundary conditions or aspiration levels. Solutions for which the critical value is below the threshold are eliminated. Consequently for assessment, the boundary conditions must be checked first.

A rating scale is a specific type of ordinal scale that encodes preferences between different objects of a set. While objects in ordinal scales are pre-ordered, objects in rating scales are ordered by the user in the rating process. The following example illustrates this: A subject in a termbank test has to rate the importance of information categories in termbanks: The set of possible categories is:

- Definitions
- Context
- Grammar
- Graphics

The answers given by the user are:

1. Grammar
2. Definitions
3. Context
4. Graphics.

The question of discreteness is important for rating scale values: can one infer information about comparisons of differences from the scales? In other words, is number 3 three times better than number 1? If two objects are rated on the same scale, and one gets 1 and the other 3, does it mean just that one object is better than another? Or, that it is somewhat or clearly better than another? Or, that it is three times better than another? French (1986:76) points to the fact that there is a great danger of reading too much into the numerical representation of both ordinal and rating scale values, since they encode only *preference order* information and are not capable of providing any idea of *strength-of-preference.* It follows that a comparison of values, particularly of mean scale values, is not considered quantitatively meaningful.

Winterfeldt/Edwards (1986:211), however, do not agree that procedures based on *strength-of-preference* judgements are more inaccurate and untrustworthy than *preference* or *indifference* judgements. They argue that *strength of preference* is a subjective magnitude like any other, for instance, *brightness of a room,* that is routinely studied by psychophysicists since Fechner (1860). As a consequence, Winterfeldt/Edwards (1986:217) conclude that utility measurement can make use of *direct rating*, a technique used in psychophysics. It requires the subject to consider at least three stimuli: two stimuli that provide end points and one that is used to elicit the numerical judgement. In direct rating, the anchors are usually a "bad" or least preferred stimulus that is arbitrarily assigned a value of 0, and a "good" or most preferred stimulus that is arbitrarily assigned a value of 100. Winterfeldt/Edwards (1986:227-229) elaborate the following example for direct rating:

A fresh M.D. considers a number of positions as an assistant professor of surgery offered in the following cities: Ann Arbor, Boston, Chicago, Los Angeles, San Francisco. The analyst first asks the decision maker to select the worst and the best city in terms of location. The decision maker instantly comes up with: worst: Chicago; best: San Francisco. The attributes on which the decision was based mainly were: climate, cultural life, size of city, pollution. After the meaning of the scale has been clarified, the other cities are ranked between the two extremes: 1. San Franciso, 2. Boston, 3. Los Angeles, 4. Chicago (reconsidered), 5. Ann Arbor. When translating this qualitative information into a quantitative value scale, the different cities are located between 0 = Ann Arbor and 100 = San Francisco.

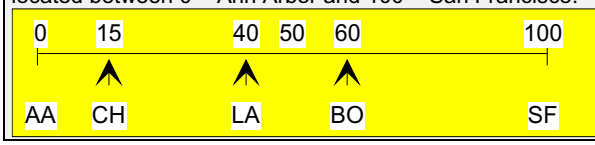


Figure 35: Sample for Direct Rating Technique

Ratio scales are characterized by

- value 1 < value 2 < value n

- equidistance

They have a fixed origin or absolute zero point and, therefore, are the only scales in which the concept "x is n-times as much as y" has any meaning. Consequently, differences in ratio scales can be compared across systems.

3.1.1.3 Measurement Issues

Scientifically speaking, measurement consists of rules for assigning numbers to objects to represent quantities of attributes. These rules must be explicitly formulated for a measure to be *valid* and *reliable*.

Validity refers to the extent to which it is possible to generalise from the circumstances of an experiment to the circumstances in real life.

Reliability concerns the extent to which measurements are repeatable - by the same individual using different measures of the same attribute or by different persons using the same measure of an attribute.

According to Nunnally/Durham (1975:311), an element of error is involved in any type of measurement, whether it is the measurement of the temperature of liquids, blood pressure or intelligence. Due to the subjectivity of psychological attributes, however, their measurement is more error prone than that of physical reality. The frequent usage of rating scales in psychological measurement, moreover, stresses the problem of *reliability*. Nunnally (1975:108) points to the major problems related to *reliability*:

- It is difficult to explicitly instruct subjects as to how to perform rating tasks. Consequently not only the data as such but also the method rests on intuition;
- There is a high variability from rater to rater and from occasion to occasion.

A cardinal way to increase *reliability* is to employ multiple raters and average their responses. Another means to increase test *reliability* is to make use of standardised measures. Standardisation is achieved if different examiners give approximately the same scores to the same subjects. Apart from increasing *reliability*, the usage of standardised measured saves money, provides more detailed information and allows comparison of tests.

The nature of the experimental design is another important issue in measurement. It is a vast subject in its own right and will only be touched here briefly. Tests eliciting psychological attributes are frequently called *quasi* or *pseudo experimental*, since there are a number of individual variables which cannot be controlled. According to Nunnally (1975:134), Nunnally/Wilson (1975:228), Edwards/Guttentag/Snapper (1975:143), central features of quasi experimental design are:

1. The usage of *comparison groups* in which one group is given a specific treatment and the other (control group) is not.
2. Observations or interviews are performed before and after the treatment

Typical sources of error in quasi experimental design are: (i) variations within tests such as motivation, stress, health of subjects; and (ii) variations between tests such as systematic differences in tests, subjective scoring, or change of attitude of subjects. Also, when involved in pre- and post-testing activities, subjects are more informed with respect what to look at in the tests, if they participated in pre-testing activities. Though quasi experimental design as well as rating scales as measurement techniques are ranked low in terms of *reliability*, behavioral scientists agree that it is often **not** possible to study psychological attributes without relying on both.

For a more detailed discussion of measurement issues see Nunnally (1975); Nunnally/Wilson (1975); Nunnally/Durham (1975); Hausen/Müllerburg/Schmidt (1987). In the context of evaluating translators' aids' systems, the nature of the experimental design and the related problems as presented above, will be considered when developing test types determining the quality of translators' aids' systems. Particularly the concept of *validity* and *reliability* as well as the problem of *subjectivity* as discussed above will be taken up again in the context of the user-oriented evaluation of translators' aids' systems.

3.1.1.4 Construction of Value Functions

In short, value functions show where the scale value of an attribute is located between the two extremes of 0 and 100, where 0 means not acceptable and 100 means as well as one could hope to do. Mapping all scale values related to an attribute onto value functions makes the different values for the different objects under evaluation comparable, which is a pre-condition for aggregating a final utility score.

The exploitation of qualitative properties of scales leads to the construction of specific types of value functions. The first question to consider when elaborating value functions is whether more of a certain attribute is always better or always worse than less. If so, the value function is called *monotone* figure . If more is always better than less the value function is *monotonically increasing* (see figure 37). If more is always worse than less, the value function is *monotonically decreasing* (see figure 39). Considering real-world evaluation problems, there are often a priori reasons to assume that the value function takes a certain shape. Although natural scales and value scales are often *monotonically* related, that relation may not be *linear*. Apart from *linear* value functions, the most frequent shapes of value functions in the context of decision

theory are *concave* and *convex*. Winterfeldt/Edwards (1986:240) illustrate with the following example how the shape of the value function can be determined by means of defining the *midpoint value*.

An executive argues: *"If I have to drive at all, the first 30 minutes of driving time to the airport are more tedious. After that an extra five minutes won't bother me much."* In other words, a fixed extra amount of driving time seems less tedious after longer initial drives than after shorter ones. This indicates that the value of driving 30 minutes must be in the lower quadrant. It also indicates that the midpoint in value must be below 30 minutes driving (left quadrant). The shape of the resulting value function is **convex**.

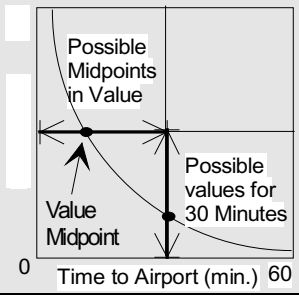
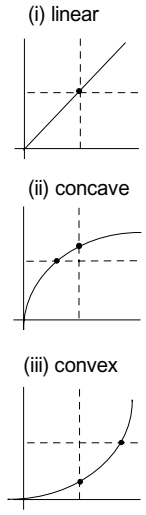


Figure 36: Curve Fitting Example

Monotonically Increasing Functions - More is Always Better than Less



The value **increases proportional** to an increasing value on the scale

Value Midpoint is in the left quadrant. Midpoint of scale value is in the higher quadrant.

Value Midpoint is in the right quadrant. Midpoint of scale value is in the lower quadrant.

Figure 37: Typical Shapes of Monotonically Increasing Functions

Winterfeldt/Edwards (1986:237) point out that among decision theorists it is a commonly acknowledged fact that in real world problems there are only few attributes that do not produce monotone value functions. It has been argued that psychological measures tend to fit the linear model. Ordinal value scales, for instance, always produce monotone value functions. In other words, a strictly increasing function *(ϕ)* has a strictly positive gradient everywhere. This means that a higher ordinal value will always result in a higher function value. Thus, for all x_1, x_2 it is true that

$$\phi(x_1) > \phi(x_2) \Leftrightarrow x_1 > x_2$$

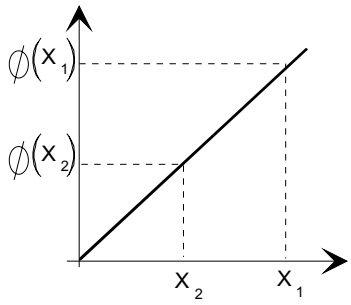


Figure 38: Ordinal Value Function

Monotonically Decreasing Functions - More is Always Worse than Less

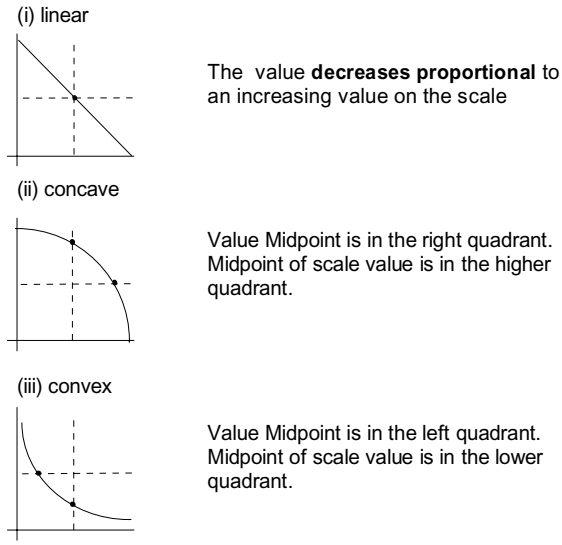


Figure 39: Typical Shapes of Monotonically Decreasing Functions

Non-Monotone Functions

If monotonicity is violated, the analyst must explore possible peaks, preference thresholds or saturation levels and adjust the function accordingly. The most important non-monotone functions are illustrated in the following figure.

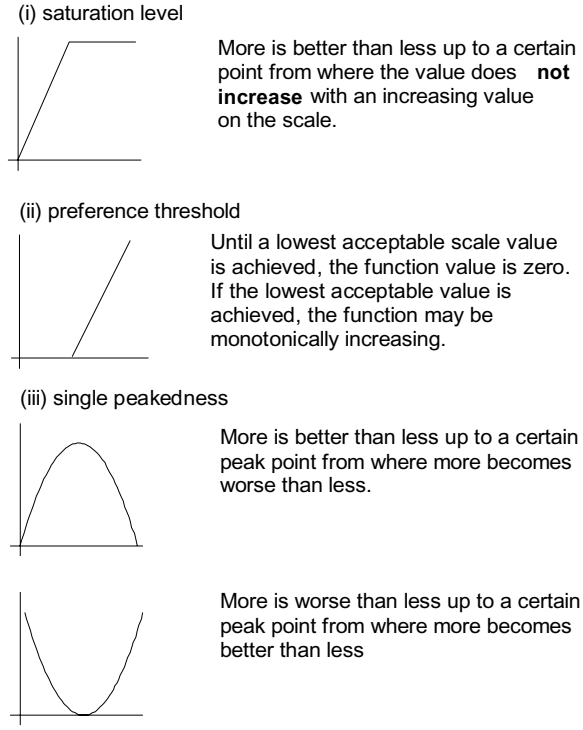


Figure 40: Typical Shapes of Non-Monotone Functions

Along the above principles, value functions can be elaborated for each attribute relevant to the decision process. Winterfeldt/Edwards (1986:258) argue that striving for precision is not the ultimate aim of the elaboration of value and utility functions. Rarely more than five points, including the two extremes of 0 and 100 have to be assessed directly. The rest can be done by curve fitting as illustrated in the figures above.

3.1.1.5 Multiattribute Utility Theory

Most decision processes involve multiple attributes that have to be first assessed individually and then combined to an overall utility score. If a decision problem involves more than two attributes, intuitive judgements of the overall value of objects is exceedingly difficult. *Multiattribute Utility Theory* explores how different values from different attributes can be combined in order to arrive at a utility score for each alternative under evaluation. Useful discussions of Multiattribute Utility Theory are presented by Keeney/Raiffa (1993); Winterfeldt/Edwards (1986); French (1986 and 1989); Hogarth (1985); Edwards/Newman (1982); Edwards/Guttentag/Snapper (1975).

In general the procedure of multiattribute evaluation involves the following steps:
1. Assigning relative weights to attributes

2. Defining aggregate utility for different options
3. Relating utility to cost and performing tradeoffs
4. Performing sensitivity analysis

The problem of multiattribute evaluation can best be demonstrated on the basis of a simplified example that was developed for this thesis. Let us assume that the decision problem which will be further exemplified in this chapter is that of purchasing one of four given software packages with specific attributes and a given price (including maintenance).

cost of option 1:	3.000 $
cost of option 2:	3.500 $
cost of option 3:	1.000 $
cost of option 4:	2.500 $

Further details of the example will be added where relevant in the following chapters.

3.1.1.5.1 Assigning Weights to Attributes

Weights capture the essence of value judgements. The most frequently used technique for assigning weights to attributes is to determine the relative importance of branches and leaves of the value tree in a two step process:

1. Weights are assessed within each of the major branches (A, B, C, D) to compare the relative importance of the attributes within each branch by dividing 1 up among the attributes of each branch.
2. Given these assignments, weights on the attribute level are obtained by multiplying through the tree. For example, A (.43) x AA (.25) = .11. The weights of the twigs of the branches (measures) again sum up to 1.

Figure 41 is the weighted value tree as it is elaborated by the decision maker in the example. A, B, C, D are top-level attributes and AA, AB ... DB are measures of these attributes, the exact nature of which is not relevant here.

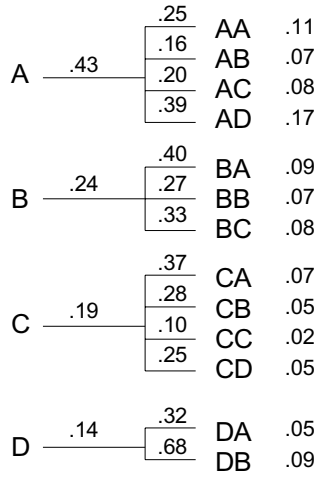


Figure 41: Weighted Value Tree

The numbers resulting from weighting process are in principle *subjective*. Different stakeholders, that is, people who have an interest in the object being evaluated, are likely to assign different weights. A possible procedure to deal with conflicting weights is presented by Winterfeldt/Edwards (1986:261) and Edwards/Newman (1982:23):

1. Discuss the individual assessments and their differences with all stakeholders concerned;
2. Perform a second weighting process;
3. Average the second weighting of all stakeholders to arrive at a final set.

Following the above procedure the weighting process is more likely to reflect the general preferences.

3.1.1.5.2 Defining Aggregate Utility

Keeney/Raiffa (1993) discuss different models for computing aggregate utility. In the context of this thesis the *additive weighted model* will be described briefly. As it was reported by Edwards/Guttentag/Snapper (1975:157) from social science research, the additive model proved to be sufficient in most practical evaluation projects. This model defines the overall value (or aggregate utility) of an option x (x = 1,2,...,n) as

$$v(x) = \sum_{i \in I} w_i v_i(x_i)$$

where $v_i(x_i)$ is the value of option **x** on the *i*th attribute;
w_i is the importance weight of the *i*th attribute; and
v is the value of **x.**

Strictly speaking the additive weighted model assumes *value independence* between attributes. To recall, *value independence* is violated, if the evaluation of an alternative with respect to one attribute depends on how the alternative performs with respect to the other. Though *value independence* is not easily satisfied, practical evaluation experience showed that in the presence of even modest amounts of measurement error, quite substantial amounts of deviation from *value independence* will make little difference to the ultimate number of **v**. As Edwards/Guttentag/Snapper (1975:157) conclude from practical evaluation experience: "A frequently satisfied condition that makes the assumption of value independence very unlikely to cause trouble is monotonicity, that is, the additive approximation will almost always work well if, for each dimension more is preferable to less or less is preferable to more throughout the range of the dimension that is involved in the evaluation, for all available values of the other dimensions."

According to Winterfeldt/Edwards (1986:309) a clever analyst can structure virtually any evaluation problem so that an additive model is appropriate, doing away with *value dependence* by restructuring processes such as combining or splitting up attribute measures, if necessary.

Considering the example, the following table shows how the calculation of the aggregate utility works for option 1, using hypothetical values for the different measures.

TWIG LABEL	WEIGHT w_i	VALUE $v_i(x_i)$	WEIGHT x VALUE $w_i v_i(x_i)$
AA	.11	90	9.9
AB	.07	50	3.5
AC	.08	30	2.4
AD	.17	90	15.3
BA	.09	30	2.7
BB	.07	40	2.8
BC	.08	80	6.4
CA	.07	20	1.4
CB	.05	30	1.5
CC	.02	20	0.4
CD	.05	50	2.5
DA	.05	60	3.0
DB	.09	70	6.3
SUMS	1.00		58.1

Figure 42: Sample for Calculation of Aggregate Utility for Option 1

To conclude, the additive weighted model provides the evaluator with means to determine the utility of an alternative under evaluation with respect to the different

attributes that are considered important in the evaluation context. There may, however, be many pitfalls in this procedure, if, for instance, attributes are not chosen well; measurement procedures are not performed adequately; or, weights are distributed insensible to preferences. In the context of the evaluation of translators' aids, care must be taken to avoid these pitfalls in order to guarantee valid evaluation results.

3.1.1.5.3 Relating Utility to Cost - The Tradeoff Problem

To make a final comparison, cost is related to utility. To take up the above example, let us assume that the calculation of aggregate utility for the further options 2, 3 and 4 result in the following overall ranking:

1. option 1: 58.1
2. option 2: 53.2
3. option 4: 48.78
4. option 3: 43.47

Considering both aggregate utility and cost of the options of the example, the following graph can be drawn where costs are plotted on the vertical axis, with less is better than more and aggregate utility is plotted on the horizontal axis with more is better than less.

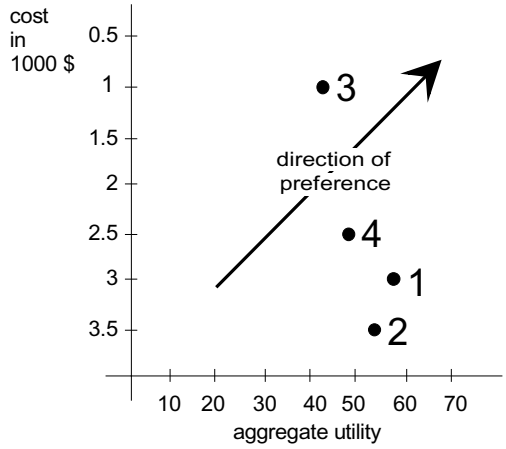


Figure 43: Graphic Representation of Utility Versus Cost

Without considering tradeoffs, the decision maker can infer from figure 43 that, as it is, option 2 is inferior to option 1, because it is more expensive and has a lower aggregate utility. For the remaining options (1, 3, 4) the decision maker has to consider how value trades off against cost. In other words, how much achievement on objective 1 is the decision maker willing to give up in order to improve achievement on objective 2 by some fixed amount? According to Keeney/Raiffa (1993:66) and

Winterfeldt/Edwards (1986:6), tradeoffs are judgements which depend on the decision maker's personal assessment of the relative desirability of the available options.

There are various ways dealing with tradeoffs. In the context of this thesis the *pricing out* procedure presented by Keeney/Raiffa (1993:pp.125) will be discussed in more detail. *Pricing out* implies that money (earnings/savings) is treated as one of the attributes in assessment on a par with the other attributes rather than only in the final stage when the other criteria have already been summed up. Including money as one of the criteria, rather than something outside evaluation underlines the point that money is not the only thing valued.

Applying the *pricing out* procedure to the example, the first step would be to establish a relationship between utility and cost in form of a value function. Let us assume that for the decision maker in the example the utility of an option decreases not proportional to cost. While the price is rather low, the decision maker does not mind too much to pay more for an increase in utility. However, once the price is above a certain point, the decision maker is less willing to pay more for an increase in utility. This results in a concave, monotonically decreasing function where

- more is always worse than less
- the value midpoint is in the right quadrant
- the scale midpoint is in the higher quadrant

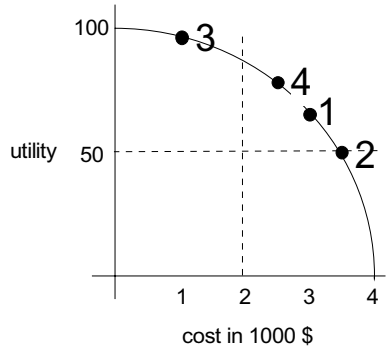


Figure 44: Value Function Relating Cost to Utility for Options (1-4)

The above figure shows where the four options under evaluation are situated on the curve. According to the *pricing out* method, money will be introduced as additional attribute **E**, with the following values for the different options: $v(1_E) = 65$; $v(2_E) = 50$; $v(3_E) = 93$; $v(4_E) = 75$.

option number	attribute A	attribute B	attribute C	attribute D	attribute E
1	71.6	50.0	30.8	66.2	65
2	48.4	50.0	64.0	59.2	50
3	43.9	47.5	13.8	75.6	93
4	63.6	60.0	32.4	6.4	75

Figure 45: Subaggregate Utilities of four Options under Evaluation including Cost as Attribute E

The next step of the *pricing out* method is to consider the weight of attribute E as compared to attributes A - D. It is important to note that in the example attribute E consists of no sub-attributes or measures. In other circumstances it is also thinkable to split up cost into smaller units such as purchase cost, maintenance, hardware etc. The pricing out methods asks that the new attribute E has to be included in the original value tree and the distribution of weights reconsidered (see figure 46).

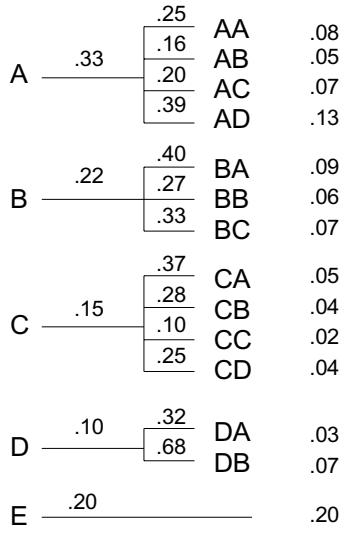


Figure 46: Weighted Value Tree including Cost as Attribute E

In a last step, the additive utility function can be applied to the value tree, leading to a final result which inherently considers cost as one of the attributes relevant to evaluation. For x = (1,2,3,4); with $w_A = .33$; $w_B = .22$; $w_C = .15$; $w_D = .10$; $w_E = .20$; the results of the analysis after Keeney/Raiffa's *pricing out* method would be as follows:

X	V_A	$W_A(V_A)$	V_B	$W_B(V_B)$	V_C	$W_C(V_C)$	V_D	$W_D(V_D)$	V_E	$W_E(V_E)$	V(X)
1	71.6	23.62	50.0	11	30.8	4.62	66.2	6.62	65	13	58.86
2	48.4	15.97	50.0	11	64.0	9.6	59.2	5.92	50	10	52.49
3	43.9	14.48	47.5	10.45	13.8	2.07	75.6		93	18.6	53.16
4	63.6	20.98	60.0	13.2	32.4	4.86	6.4	0.64	75	15	54.68

Figure 47: Utility Measurement for Options 1,2,3,4 in Pricing Out Model

Making use of the *pricing out* method, the options would rank as follows:

1. option 1: 58.86
2. option 4: 54.68
3. option 3: 53.68
4. option 2: 52.49

Once the analyst has arrived at a ranking of options in terms of both cost and utility along the procedure presented above, it has to be made sure, that the analysis process is without major flaws and actually represents the views of the decision maker. This final phase of the decision analysis process is called *sensitivity analysis*.

3.1.1.5.4 Performing Sensitivity Analysis

The most important function of sensitivity analysis is to give the client confidence in the robustness of the analysis. Sensitivity analysis is not special to decision analysis. In the context of operations research and management science, computer programs help to perform sensitivity analysis. In the context of this thesis the discussion will be reduced to its most basic principles. A more detailed discussion can be found in Winterfeldt/Edwards (1986:388-401); Edwards/Newman (1982:81-92).

Sensitivity analysis consists of changing some of the numbers that went into the multiattribute utility calculation. Most sensitivity analyses are one-dimensional, that is, they vary one parameter at a time. The most important kind of sensitivity to look at is the sensitivity to *weights*, since weights are purely subjective numbers about which people tend to disagree. In short, for this purpose the evaluator has to consider the values of the individual attributes for all alternatives under evaluation a priori to weighting and compare this to the weighted figures. If there is an unexpected discrepancy the evaluator can change the weights of individual attributes and consider the extent to which the change of weights affects the ranks of the alternatives under evaluation. The utility calculation can be considered robust, if minor changes of weights do not affect the ranking of alternatives.

Apart from considering the sensitivity to weights, the reliability and validity of measures can be reconsidered during sensitivity analysis. Subject to examination could be measures that (i) have a high overall weight in the value tree; or (ii) deliver a broad range of values between options. The questions to follow at this stage are:

- Does the measure actually make sense or should it be reformulated?
- Are the results repeatable?
- Was the experimental design without flaws?
- Are there any calculation errors?

The decision analysis process can be said to be finished successfully, if sensibility analysis either showed the robustness of the original analysis or could detect and correct flaws.

To conclude, the procedures used in decision analysis may have an important impact on the development of evaluation and assessment procedures of translators' aids' systems, since it is concerned with measuring and combining values related to different attributes. It may allow the comparison of the suitability of different translators' aids' systems for a particular work context on a quantitative basis.

3.1.2 Approaches from Software Engineering

The ultimate aim of structuring approaches in software engineering lies in detecting and defining properties software systems must have in order to be accepted by clients, and not in quantitatively measuring the extent to which the system fulfils requirements. The discipline of requirements engineering presents a great number of highly formalized modelling approaches which are used to structure the domain so as to deliver important input to system specification. In section 3.1.2.1 only those approaches will be outlined, that have some relevance for the evaluation of translators' aids' systems. In section 3.1.2.2 the principles of quality requirements definition, which is used to define how systems should ideally behave in order to fulfil user requirements, are described.

3.1.2.1 Modelling Approaches in Requirements Engineering

Sommerville (1996:99-115) provides a broad overview of modelling techniques, which mainly differ in their focus and underlying principle. In the context of this thesis attention will be focused on the description of mainly three approaches to modelling:

1. *Generic Task Modelling* is concerned with getting domain expertise into representations usable for subsequent design. In the context of evaluation it can be used to identify those issues of the translation task that are relevant to the elaboration of measurable primitives. Applying generic Task modelling procedures to the translation task in evaluation, the effort of the elaboration of metrics will be reduced, since, if parts of tasks are generic, the metrics measuring them will then be generic as well.
2. *Structured Analysis* is concerned with the separate description of a system's functions and data. In evaluation it will lead to the elaboration of measurable primitives for those functions and data of translators' aids' systems that are under testing. It will serve as means to understand the relationship between the

different aspects involved in the translation process, which is to be supported by the computer.

3. *Object-Oriented Modelling* is concerned with the description of objects, their inherent information and behaviour as well as the operations performed on these objects. It will be used to elaborate metrics on the basis of an understanding of the inter-relationships of objects and operations involved in the overall translation process. It will also help in the definition of testing scenarios.

3.1.2.1.1 Generic Task Modelling

For the evaluation of translators' aids, the description of the translation task plays a central role. In the software development context the principal objective of generic task modelling is to guide the transformation of user requirements into design specifications, that is, to define implementable primitives. For evaluation purposes, generic task modelling should lead to the definition of measurable primitives. On a general level, task analysis is concerned with the questions: what is a task, how comprehensive is it, when does it terminate? There is some consensus that tasks can be seen and defined within a four-level hierarchy of project, tasks, subtasks and activities. There is a vast amount of literature on methodologies for task analysis at hand, which mainly differs in terms of focus, formalisation, and presentation. For the exhaustive discussion of the topic see Diaper (1989-1,2,3) and (1990); Hewitt/Hobson/Sapsford-Francis (1990); Wilson (1989); Carroll/Grudin/McGrew/Scapin (1990); Ip/Damodaran/Olphert/Maguire (1990); Johnson/Johnson (1990); Sewell (1990); Dzida/Freitag/Hoffmann/Vlader (1990); Sharratt (1990). What all methodologies have in common is the principal procedure, which can roughly be summarized, in the following four steps:

1. Elicitation: by means of different elicitation techniques details of the working context are elicited;
2. Description: specific actions carried out by the user and specific objects handled by the user are described;
3. Grouping: similar actions are grouped to establish subtasks and tasks;
4. Generification: from specific subtasks and tasks more general task descriptions are elaborated.

The task modelling approach developed by Dan Diaper (1989 and 1990) seems to be the one that is most appropriate to lead to the definition of measurable primitives in the context of the evaluation of translators' aids' systems. TAKD (**T**ask **A**nalysis for **K**nowledge **D**escription) consists of a methodology that describes the generation of a hierarchical description of tasks and a knowledge representation grammar. While the

representation grammar may be useful in the context of software specification, its degree of formalisation is not needed in the context of evaluation. Focus, therefore, will be put on the elaboration of task hierarchies. The TAKD methodology takes as input an *activity list*, which contains a prose description of activities carried out by an observed task performer. Data is extracted from this description in the form of *specific objects* and *specific actions*. Specific objects may be physical objects (mouse etc.) or informational objects (files, programs etc.), which are grouped into a *specific object list*. Specific actions denote the activities performed by a person that are directed towards specific objects. According to Diaper (1989-2:114) there are surprisingly few different specific actions that people carry out in computing tasks. In most cases similar actions are repeated on different specific objects. For each specific action the related objects are noted and a hierarchy of actions is developed. Similar actions are then grouped together and a generic description of a task developed from it. The resulting generic actions can be represented in form of generic statements.

3.1.2.1.2 Structured Analysis

A central element of structured analysis is the dataflow diagram. It allows the presentation of a system as a network of functional processes. In the context of evaluation, the dataflow diagram may be useful to develop metrics from the description of the functional processes. It may further be used to make sure that the context of the functions under testing is considered properly when preparing tests of translators' aids' systems, and thus make sure that the tests can be applied successfully. Yourdon (1989:pp.139) describes the components of a dataflow diagram as follows:

> The *process* shows a part of the system that transforms inputs into outputs. It is graphically represented by a circle, named or described with a word, phrase or simple sentence that denotes the nature of the process.
>
> The *dataflow* is used to describe the movement of information from one part of the system to another. It is graphically represented by an arrow into or out of a process, into or out of a store, into and out of a terminator. If possible, the kind of information transferred is named. One may distinguish between input flows, output flows, dialogue flows, and diverging flows.
>
> The *data store* is used to model a collection of data at rest. Stores are typically named with the plural of the name of the information that is carried by flows into or out of the store. A flow from a store is normally interpreted as a read, or an access to information in the store. A flow to a store is often described as a write, an update, or possibly a delete of information in the store.
>
> The *terminator* represents external entities with which the system communicates. Typically, a terminator is a person or group of people outside the control of the

system being modelled, or also, in some cases, some other computer system with which the system modelled will communicate. It is graphically represented by a rectangle.

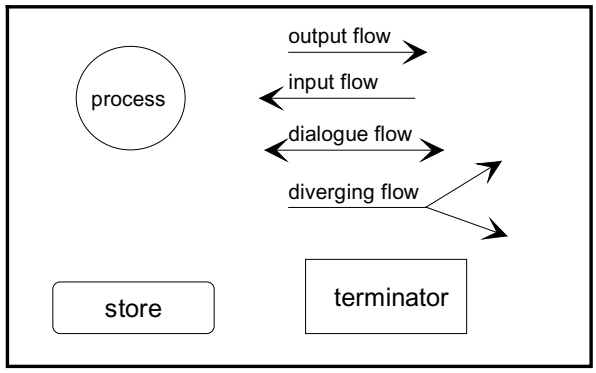


Figure 48: Components of a Dataflow Diagram according to Yourdon (1989:pp.139)

In the context of software development the dataflow diagram is supplemented by data dictionaries, which list the data elements pertinent to the system; process specifications, which describe in detail what is happening inside each process; and entity-relationship diagrams, which describe the relationship between the stored data. While for software development this attention to detail may be necessary, the evaluation of translators' aids' systems does not ask for this level of description.

3.1.2.1.3 Object-Oriented Modelling

Object-oriented models consider a system as a number of interacting objects. Jacobson (1992:465–493) describes a great number of object-oriented methods, which can be distinguished mainly with respect to their object types; formalisation of methods; degree of formalisation; and user orientation. For the purpose of user-oriented evaluation of translators' aids' systems, the *use case model,* as described by Jacobson (1992:128-132) is of particular interest. It may be used to elaborate metrics on the basis of an understanding of the inter-relationships of actors, objects and operations involved in the usage of translators' aids' systems. It may also help in the definition of different testing scenarios.

The *use case model* uses *actors* to represent the roles users (human or other systems) can play and *use cases* to represent what users should be able to do with the system. Actor is a class and is defined to be everything external to the system with which the system communicates. Different users are instances of the class actor. Each actor will perform a number of use cases to the system. A use case is a special sequence of related transactions performed by an actor and the system in a dialogue. Each use case

is a specific way of using the system and every execution of the use case may be viewed as an instance of the use case. The following questions will help to identify use cases:

- what are the main tasks of each actor?
- will the actor have to read/write/change any of the system information?
- will the actor have to inform the system about outside changes?
- does the actor wish to be informed about unexpected changes?

3.1.2.2 Quality Requirements Definition

The process of quality requirements definition leads to a number of quality characteristics of software which denote what is generally considered as good characteristic of a software system. Today's approaches to quality requirements definition largely go back to the early models by Boehm/Brown/Lipow (1976) and McCall/Richards/Walters (1977). Further exhaustive examples for different quality models can be found in Boehm et al. (1978); Christ et al. 1984; Murine (1983), and (1986); Gaines (1987); Höge/Wiedenmann/Kroupa (1991). Quality requirement models make use of a top-down approach, starting with the top level concept of quality, and splitting it up into a varying number and varying levels of quality attributes. For each quality attribute that cannot be split up into further attributes, metrics have to be developed. McCall/Richards/Walters (1977:2.2) define metrics as measures of the attributes related to the quality characteristics. Most quality models focus on different quality characteristics and provide different definitions and implication schemes. There is also much dissent concerning the principal terminology used for the decomposition of quality. Whereas some stick to the terms *quality factors* and *criteria*, others prefer *quality characteristics* and *attributes*. The International Organisation for Standardisation (ISO) developed a standard for the definition of quality requirements in 1991, ISO 9126, which will be used in this context.

ISO 9126 decomposes the general concept of quality into six quality characteristics and twenty-one subcharacteristics. For each area of application those quality characteristics and subcharacteristics need to be identified which are of particular importance and metrics have to be developed accordingly.

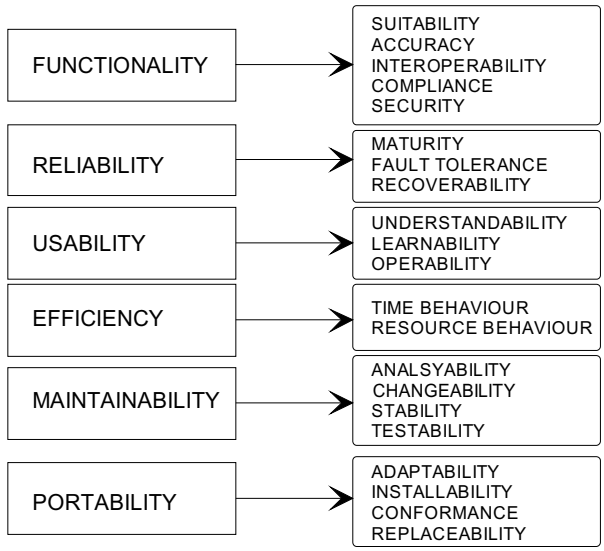


Figure 49: ISO (1991:4.1-6) Quality Decomposition

The ISO definition of each of the above quality characteristics will be presented and discussed in the light of translators' aids' systems. It will be demonstrated how it is possible to arrive at metrics when applying the software quality characteristics to the translation task. According to Sommerville (1996:118) and many other software engineers it is useful to distinguish between **functional** and **non-functional** requirements**.** Roughly, functional requirements are system services which in the ISO model fall under the quality characteristic functionality, while non-functional requirements set out constraints under which the system must operate.

The Quality Characteristic functionality

Functional properties or functional requirements, as they are called in the requirements engineering context, are related to the type of tasks users perform with the system, that is, in this context, those involved when translating documents. The quality characteristic *functionality* covers all aspects that are relevant in order to perform these tasks. ISO 9126 defines *functionality* as follows:
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

Functionality determines how well a system {M} can accomplish given tasks {D}. The system function can be taken as a specialisation of the task, that is, a way to perform a task, and consequently is not ontologically different from the task as such.

In chapter 1, conclusions were drawn with respect to the required functionality of translation systems from

(i) the types of tasks that are performed during the translation process;
(ii) the types of problems encountered; and
(iii) the types of strategies that are generally employed to solve them.

To recall, one of the source text analysis tasks, for instance, is *determination of extralinguistic reference*. The following figure shows how this task led to the determination of the required function.

TASK	PROBLEMS	STRATEGIES	FUNCTIONS
determination of extralinguistic reference	sufficient knowledge on the subject matter of the source text	text documentation in SL	on-line text corpus
	comprehension of the details of the source text	terminology look-up	termbank

Figure 50: Example for Determination of System Functions from Translation Tasks

Following the same approach, a number of central functions or modules for translators' aids' systems were identified in the same chapter, that is

- identification of repetitions in texts and retrieval of translations
- retrieval of encyclopaedic information
- full text retrieval
- terminology database
- terminology elaboration

The above modules represent the "set of functions" that are relevant to all translators' aids' systems to a certain extent. It can be taken as a **functional skeleton** of translators' aids' systems. The required properties of the functions may be different from user group to user group. To elaborate the specific "properties" of each of the above functions for the specific usage context, the different tasks performed by the translator have to be considered in view of the subcharacteristics of *functionality* as defined by ISO 9126, that is, *suitability, accuracy, interoperability, compliance, and security.*

suitability	Attribute of software that bears on the *presence* and *appropriateness* of a set of functions for specified tasks.
accuracy	Attributes of software that bears on the provision of *right or agreed results* or effects.
interoperability	Attributes of software that bear on its *ability to interact* with specified systems.
compliance	Attributes of software that make the software *adhere to* application related *standards* or conventions or regulations in laws and similar prescriptions.
security	Attributes of software that bear on its *ability to prevent unauthorised access*, whether accidental or deliberate, to programs and data.

Figure 51: Subcharacteristics of *functionality* according to ISO 9126

For each task performed by the translator in the specific context, the evaluator has to determine the strategies and required attributes. Each of the attributes relevant to the translator then can be considered in view of the subcharacteristics of *functionality*, that is, *suitability*, *accuracy*, *interoperability*, *compliance* and *security*. The following table demonstrates this procedure for the task *determination of extralinguistic reference*:

STRATEGIES	ATTRIBUTES	SUB-CHARACTERISTIC	PROPERTIES
terminology look-up	definitions of terms in SL	SUITABILITY	• retrieval module • definition in SL as info category • definition useful for problem • concept structures (thesaurus) included
		ACCURACY	• definition correct
		INTER-OPERABILITY	• can be accessed from text processing
		COMPLIANCE	• standard terminology interchange format
		SECURITY	• password check • read/write access rights
	concept structures	SUITABILITY	• thesaurus • sub- and superordinated concepts
		ACCURACY	• type of concept relation specified (part/whole, effect etc.)
		INTER-OPERABILITY	• can be accessed from text processing
		COMPLIANCE	• existing thesauri • standard relationships (part/whole, effect etc.)
		SECURITY	• read/write access rights

Figure 52: Specification of Properties of Functions from Task

To summarise, the specification of the *functionality* of translators' aids' systems can be based on the **tasks** that are performed during the translation process. Considering the translation tasks in the light of the ISO subcharacteristics of *functionality* leads to

the determination of **functional properties** of these tasks. According to ISO 9126 (4.1), these properties have to "satisfy the stated or implied needs" of translators. The needs of translators, however, may vary considerably in different situations. Consequently, one of the sub-characteristics of functionality should reflect the extent to which a system allows its adjustment to specific needs by customers themselves. In other words, the performance of many translators' aids' systems strongly depends on the possibilities the systems offer to put particular constraints on the processing of system inputs or the elaboration of data resources. For translation memories, for instance, one and the same fuzzy matching algorithm will deliver totally different results, if one system offers the possibility to define certain strings that are to be used as variables (e.g. product names, versions of products, company names etc.) during the database search and the other system does not. Therefore, for translators' aids' systems, each system should also be considered in terms of its *customisability*, leading to the following add-on of subcharacteristics of *functionality*:

Customisability	Attributes of software that allow the user to set specific constraints on a systems' inputs, processing of inputs and data resources.

Figure 53: *Customisability* as Additional Subcharacteristic of *Functionality*

In the above example for *determination of extralinguistic reference*, *customisability* could lead to the following properties:

STRATEGIES	ATTRIBUTES	SUB-CHARACTERISTIC	PROPERTIES
terminology look-up	definitions of terms in SL	CUSTOMISABILITY	• retrieval of definitions with constraints: e.g. • only specific authors; • only after certain input date;
	concept structures	CUSTOMISABILITY	• retrieval only of concepts in specific subject area

Figure 54: Specification of *Customisability*-Properties from Tasks

The above examples demonstrated how functional properties of systems can be elaborated from a mapping of tasks to quality characteristics. In a next step properties or attributes need to be mapped onto scales in order to allow the measurement of the extent to which a property has been fulfilled by a system under evaluation.

Starting from the definitions of the subcharacteristics of *functionality* as presented above, measurement specific issues need to be elaborated. Referring to the definition of *suitability*, the "presence" of features can be measured on nominal scales, by

transforming n-valued nominal scales into binary nominal scales. Binary nominal scales are used in checklists. Measuring the "appropriateness" of a set of functions is much more complex than measuring their presence. The appropriateness of a function can be measured on

(i) ratio scales, if a numerical value can be specified as result of the measurement process (e.g. number of entries in a termbank);
(ii) ordinal scales, if the preferences of system behaviour are determined subjectively by decision makers (e.g. preference of one solution over another);
(iii) binary scales, if the result of a measurement procedure is either that a function does or does not what it is supposed to do (e.g. retrieve only terms of a specific subject area).

The major problem of developing metrics for appropriateness is when using ordinal scales. Ordinal scales are artificial constructs to elicit subjective aspects of quality. The design of the scale determines the types of answers that can be given. The most frequently used types of ordinal scales offer 5 degrees where strength of preference can be specified. When developing ordinal type metrics one has to be aware of the problems of subjectivity and discreteness when comparing results obtained by means of ordinal scales.

Metrics for *accuracy* ask for measuring "right or agreed results or effects", which can be done on

(i) binary scales, if there is a clear definition of right and wrong (e.g. a FL equivalent in termbanks);
(ii) nominal scales, if certain nominal values can be specified as required properties (e.g. information on translator, change date etc., coming up with TM retrieval);
(iii) ratio scales, if numerical values are the outcomes of measurement operations (e.g. number of hits in termbank);
(iv) ordinal scales, if a threshold can be defined which the results of the measurement have to surpass (e.g. acceptability of translation proposal)

Interoperability involves measuring the "ability to interact with specified systems," which can be performed on

(i) nominal scales, if systems can be specified with which interaction is wanted (e.g. editor interaction with termbank, translation memory and MT system);
(ii) binary scales, if the possibility to interact with a specific system is questioned (e.g. interaction with CD ROM machine translation system possible?)

Measuring *compliance* asks for investigating, whether systems "adhere to standards". This can be performed on binary scales, whenever a comparison is possible between

the system under testing and the standard (e.g. window for opening TM databases the same as opening files in Windows?). Similarly, for *security*, metrics for assessing the "ability to prevent unauthorised access" can be measured on a binary scale, whenever an interaction requires read/write control (e.g. modifying a termbank).

Metrics measuring *customisability* consider the "constraints" that can be set on specific functions. This can be done using

(i) binary scales, if the existence of a specific constraint is checked (e.g. possible to retrieve only terms by specific author?)
(ii) nominal scales, if a set of constraints is required (e.g. TM retrieval module with variables for time, date, brand names, and type names?)
(iii) ratio scales, if the number of constraints that the system offers is relevant (e.g. insertion of project names, project numbers, order numbers in termbank entry possible?)

Non-Functional Quality Characteristics

While functional quality characteristics are closely related to the type of software system, that is, in this case a language engineering system, non-functional quality characteristics are equally relevant to any type of application software. The depth of the definition of non-functional properties depends on the role a non-functional quality characteristic plays in a specific evaluation process. Before elaborating non-functional properties, the evaluator has to determine the role of the non-functional characteristic for the specific analysis process. There is a great danger that the evaluation procedure becomes *unmanageable*, if too many non-functional characteristics that play minor roles are considered in the final decision process.

Under non-functional quality characteristics will be subsumed the ISO characteristics *reliability*, *usability*, *efficiency*, *maintainability*, *portability*. The following table outlines the ISO 9126 (4.2-4.6) definitions for the above characteristics:

reliability	A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
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.
efficiency	A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
maintainability	A set of attributes that bear on the effort needed to make specified modifications.
portability	A set of attributes that bear on the ability of software to be transferred from one environment to another.

Figure 55: Non-Functional Quality Characteristics

Each software function or module can be evaluated to some extent in terms of its *reliability*, *usability*, *efficiency*, *maintainability* and *portability*. It is important to note that while the functional properties of a translators' aids' system can be determined mainly on the basis of a task-description {D}, non-functional properties of translators' aids' systems should take the system context {M} into consideration, otherwise inadequate preferences may be determined. For instance, when considering the *operability* of a system, it is inadequate to specify that buttons are preferred to menus, since one system may have a quicker menu solution than another may have a button solution. In the following the different non-functional quality characteristics will be discussed briefly.

Reliability

ISO 9126 (A.2.2.1-2.2.3) subsumes *maturity*, *fault tolerance* and *recoverability* under the quality characteristic of *reliability*.

maturity	Attributes of software that bear on the frequency of failure by faults in the software.
fault tolerance	Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringements of its specified interface.
recoverability	Attributes of the software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.

Figure 56: Subcharacteristics of *Reliability*

Maturity is a quality characteristic that is typically of great relevance in the software development process. Accordingly, *maturity* is one of the most basic quality subcharacteristics for progress evaluation, since the goal of progress evaluation is to show an improvement of the software between two stages of development, mirrored (amongst other things) by a reduced frequency of failures by faults in the software. The "frequency of failure" can be measured on

(i) ratio scales, if the number of failures is counted (e.g. number of system breakdowns during operation);
(ii) interval scales, if the time between the failures is measured (e.g. standard metric MTBT mean time between failures).

Fault tolerance is concerned with both "software faults" and "infringements" of its specified interface. For user-oriented evaluation *fault tolerance* concerns aspects of communicating with the system. In general, it has to be assessed whether the software can cope with what is passed to it, either through programmatic interfaces or through the user interface (the operating system converts user interface events into programmatic calls to the software). A *fault tolerant* system should perform input

check routines and should not die when erroneous data is passed. On the user interface side, there is another aspect of *fault tolerance*, which should not be neglected, that is, in case of user errors, for instance when the user performs an unintended action. In this case it has to be investigated whether there are "cancel" or "undo" options for user-initiated processes. The "ability to maintain a specified level of performance" can be measured on

(i) binary scales, if it is examined whether the system, after a fault has occurred, takes up processing or not (e.g. can TM be opened after breakdown?);
(ii) nominal scales, if it is examined what options there are to maintain performance after a fault (cancel, undo, and escape options?);
(iii) ratio scales, if the effort is measured to maintain the level of performance (e.g. counting the keystrokes necessary to undo an unintended operation).

The above definition of *recoverability* is threefold, that is, it investigates (i) whether the system will take up processing after a failure, (ii) whether the data that was being processed when the failure occurred is still correct and consistent, and (iii) what effort is needed to get system and data back to normal. Recovery tests are typically performed whenever a great amount of data is processed, and various data sources are accessed (e.g. batch translation making use of MT, translation memories, termbanks, checking tools). The extent to which data (input as well as output) can be recovered correctly and consistently has to be investigated. Other aspects are whether the different modules affected deliver correct and consistent data after recovery, and which actions are necessary to get things back to normal. Metrics measuring the "capability" as well as the "time and effort to re-establish a specific level of performance" are mapped onto

(i) binary scales, if it is examined whether the performance can re re-established at all (e.g. are modifications of termbank saved after system breakdown?);
(ii) nominal scales, if it is examined what options there are to re-establish a level of performance (cancel, undo, and escape options?);
(iii) ratio scales, if the effort is measured to re-establish the level of performance (e.g. time needed to re-enter the modifications in a termbank).

Usability

According to ISO 9126 (4.3) users may include operators, end users and indirect users who are under the influence or dependent on the use of the software. For evaluation purposes it is important to make clear whose point of view is considered when defining the target properties of the system under evaluation. According to ISO *Usability* must address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results. ISO 9126

(A2.3.1-2.3.3) subsumes the characteristics *understandability*, *learnability*, and *operability* under *usability*.

understandability	Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability.
learnability	Attributes of software that bear on the user's effort for learning its application (for example, operation control, input, output).
operability	Attributes of software that bear on the users' effort for operation and operation control.

Figure 57: Subcharacteristics of *Usability*

As the above table shows, in the ISO standard, the key issue of all subcharacteristics of *usability* is "effort." For systems, which produce or offer textual information, *understandability* can have a very high priority. The "effort for recognising the logical concept" can be measured on different levels, making use of different scales:

(i) binary scales, if it is examined whether the concept can be understood at all (e.g. functionality of translation memories);
(ii) ratio scales, if the effort to understand the system is measured (e.g. frequency of calling help);
(iii) ordinal scales, if the extent to which a concept can be understood is questioned (e.g. understandability of interface layout).

Learnability is closely linked to both *understandability* and *operability*: if a system is understandable, the user can memorise more easily what kind of input or action is needed in order to solve a particular task; if a system rates high with respect to *operability*, that is, offers a well designed and flexible user interface, the user will learn quickly how to interact with the system. Metrics for measuring the "effort for learning an application" can be mapped on the following scales:

(i) ratio scale, if it is examined how much help is needed (e.g. hours of training programme);
(ii) nominal scale, if it is examined what type of help is used (e.g. training programme, documentation, on-line help, user hotelmen etc.).

The primary focus when evaluating a system's *operability* lies on the assessment of the user interface, that is, the extent to which the user interface supports the tasks that need to be undertaken with the software. The effort of handling given user interfaces, however, is not objectively assessable. It is based on likes and dislikes of individual software users, who may have a different computational background: someone who is not used to handling the mouse will be quicker using key combinations for operation control, while for experienced users of windowing systems, key combinations will not always be acceptable. Using subjective criteria as the starting point in evaluating *operability*, and asking users what they like and how much they like it, therefore, is

wrong-headed. One must ask what they want to accomplish and how they can do it, and deduce preferences from that. This is particularly important, since effort also depends on the type of task, e.g. for tasks that mainly involve using the keyboard (e.g. typing), key combinations are often quicker, since one does not have to move the hand to the mouse for operation control; for tasks that involve the usage of the mouse to a great extent (e.g. drawing), direct manipulation will be quicker. Measuring **"**effort for operation" the following scales can be used:

(i) ratio scale, if the effort can be numerically assessed (e.g. number of keystrokes to achieve a task);
(ii) nominal scale, if the system can be examined along a pre-existing checklist (e.g. key-shortcuts, macros, etc.);
(iii) ordinal scale, if the strength of preference is examined (e.g. font size readable: easy ---- unreadable);
(iv) binary scale, if the existence of individual possibilities for operation control is examined (e.g. change of font size possible?).

Efficiency

In our current technological era that is characterised by performance in terms of speed and capacity, *efficiency* naturally has become more and more important in the evaluation of software systems. The most basic reason for integrating new software components into existing environments is to improve the *efficiency* of processes, that is, increasing quantity (and possibly quality) while keeping the cost factor constant (or possibly decreasing).

ISO 9126 (A.2.4.1-2.4.2) identifies two major factors that determine the *efficiency* of systems, that is, time and resources. While the time factor is of growing importance, the resource factor is less critical today than it used to be some years ago. This is due to vast developments in hardware environments (processors and memory).

time behaviour	Attributes of software that bear on response and processing times and on throughput rates in performing its function.
resource behaviour	Attributes of the software that bear on the amount of resources used and the duration of such use in performing its function.

Figure 58: Subcharacteristics of *Efficiency*

Time behaviour is a function of quantity of data processed and processing power: quantity of data processed is negative implicative, while processing power is positive implicative. While *time behaviour* can be most easily assessed for batch programs, it is impossible to define objective metrics for the *time behaviour* of an interactive system. The more system behaviour depends on user interaction or input, the more difficult it is to define units that can be measured objectively.

The *resource behaviour* of software system depends to a large extent on the complexity of programs (e.g. whether performing simple pattern matching or building up neuro networks) and the way the function is programmed: one and the same function could require different resources when being programmed by different programmers with different programming skills. Metrics of *efficiency* are mapped onto ratio scales, since they are concerned with quantities of time and resources. Typical examples for translators' aids' systems are

- response time for database queries;
- response time for batch alignment program;
- MB RAM needed for application etc.

Maintainability

From the ISO 9126 definition of *maintainability*, one can conclude that it does not play a central role in the user-oriented evaluation of translators' aids' systems, since it is mainly concerned with the effort for modifications to existing software systems which is relevant to system developers only. ISO 9126 (A2.5.1-2.5.4) subsumes the following subcharacteristics under *maintainability*:

analysability	Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.
changeability	Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.
stability	Attributes of software that bear on the risk of unexpected effect of modifications.
testability	Attributes of software that bear on the effort needed for validating the modified software.

Figure 59: Subcharacteristics of *Maintainability*

Portability

The importance of *portability* is steadily growing, since the computer has entered nearly all areas of life, confronting largely an audience with non-computational background. While some years ago, it could be expected from a computer user that he/she is at least acquainted with the principal knowledge of the then current operating system (mainly DOS), today's user does not want to be bothered with anything that goes beyond confirming or rejecting system messages. Thus all problems that are related to the integration and usage of new software in a specified individual environment need to be supported largely by today's systems. ISO 9126 (A.2.6.1-2.6.4) subsumes the following characteristics under portability:

installability	Attributes of software that bear on the effort needed to install the software in a specified environment.
conformance	Attributes of software that make the software adhere to standards or conventions relating to portability.
adaptability	Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.
replaceability	Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.

Figure 60: Subcharacteristics of *Portability*

The primary requirement related to the *portability* of systems is that a system must work after installation. Both *installability* and *conformance* have a direct effect on this requirement: if too much effort or knowledge is needed for the successful installation, in many cases an installation will fail and the user will not get the system running; in turn, the source of many installation problems is a lack of conformance of the system to be installed with those systems that are already running. Another, probably secondary, requirement related to *portability* is that the system supports the adaptation to the specific environment, which increases the usefulness of the system in that same environment. Finally, a more sophisticated requirement related to *portability* would be that the system could be used instead of other, already operable, systems in that same environment. Accordingly, a system's behaviour with respect to its *installability* and *conformance*, its *adaptability* to different environments, and, finally its *replaceability* is a decisive factor influencing the final user acceptance. The effort to install can be measured on

(i) ratio scales, if the effort can be numerically assessed (e.g. time of installation program);
(ii) binary scales, if the success of the installation is questioned (e.g. installation successful?).

"Adherence to standards" can be measured on

(i) binary scales, whenever a comparison is possible between the system under testing and the standard is possible (e.g. windows messages used for installation?);
(ii) ratio scales, if the number of violations can be counted (e.g. number of unexpected results of installation).

The "opportunity for the adaptation" of the system to another environment can be measured on the following scales:

(i) binary scale, if the possibility of adaptation is assessed (e.g. adaptation of termbank from one-user to multi-user possible?);
(ii) ratio scale, if the effort of adaptation can be numerically assessed (e.g. time spend to adapt system)

Finally, the "opportunity of using one system instead of another" can be assessed on a binary scale (e.g. exchange of IBM Text alignment tool for Trados possible?). The effort can be measured on a ratio scale (e.g. time necessary to adapt environment?).

3.1.3 Discussing Evaluation in the Light of Software Engineering and Decision Analysis

The presentation of approaches from the disciplines of decision analysis and software engineering showed that there are interesting overlaps between these two rather different disciplines that can be exploited in the context of the evaluation of translators' aids' systems. Both software engineering and decision analysis make use of very interesting methods to structure of domain problems. Decision analysis moreover delivers the baseline for measuring attributes and presents processes that allow the comparison of different systems under evaluation.

In short, user-oriented evaluation of software systems can benefit from decision analysis in various ways:

- the structuring of values in decision analysis throws an interesting light on the process of how to develop attribute hierarchies for translators' aids' systems;
- scale construction issues in decision analysis can be applied with measurements in software tests;
- the discussion of problems of subjective measurement is of fundamental importance to evaluation which centres around user problems;
- the presentation of measurement issues in decision analysis underlines the problems that are also faced in user-oriented software testing;
- the elaboration of value functions in decision analysis may provide means to relate actual test results to target values; and,
- multiattribute value theory may be used to assess the performance of one or more software systems in numerical terms.

Similarly the evaluation of translators' aids' systems can apply the modelling approaches used in software engineering. They were originally geared to defining software characteristics as input into the software specification document but can be used for reducing the effort of developing metrics for evaluation. Describing domain problems in form of models serves to make sure that one truly understands the many facets of the problem under investigation. Software engineering defines modelling as the activity of describing, classifying and categorising domain problems. In the context of evaluation, focus must be put on how to exploit the principles of classification and categorisation. The basic idea behind classification and categorisation is that problems belonging to the same class or category *share central properties*. Consequently, in

evaluation, if different problems share central properties, they can also share the metrics to measure these properties to a certain extent. In other words, modelling should be used in evaluation to reduce the effort to develop metrics.

Mapping the software quality characteristics discussed in section 3.1.2.2 onto the different modelling concepts presented in 3.1.2.1, that is, *actions*, *objects*, *actors*, *use cases*, *dataflow*, and *data*, a great number of qualitative aspects can be identified which can form the starting point for the development of metrics for any user-oriented evaluation procedure. In the following, each of the modelling concepts will be considered in the light of the different quality characteristics, and aspects that determine these quality characteristics will be listed. It has to be noted that the following lists of qualitative aspects denoting the properties of the different modelling concepts, mainly consider those quality characteristics that are particularly relevant to user-oriented evaluation, that is, *functionality*, *usability*, *reliability* and *efficiency*.

ASPECT RELATED TO **ACTIONS**	QUALITY CHARACTERISTIC
processes involved to perform action different options to perform action appropriateness of processes for actions suitability of outcome of action objects handled during action data accessed during action type of input necessary to perform action type of constraints on action interaction with other actions interaction with objects/functions similarity of actions security of actions type of result as output of action customisation of actions	functionality
effort to perform action different ways to perform action difficulties in performing actions difficulties in understanding actions typical sequence of actions help necessary during performance of action	usability
possibilities to undo actions failures during performance of actions types of failures possibility to stop actions	reliability
time needed for action resources needed for action correctness of action output	efficiency

Figure 61: Qualitative Aspects Related to Actions

Having decided, which actions are relevant to the evaluation process, the evaluator simply selects those qualitative aspects relevant to the specific evaluation process and elaborates scales for measuring the aspects. The same is true for all of the following

modelling concepts. An example of how to use these qualitative aspects of *data* for the development of metrics will follow the lists of the different modelling concepts.

ASPECT RELATED TO **OBJECTS**	QUALITY CHARACTERISTIC
function in which it is involved characteristics of object type of object size of object operation modes appropriateness of object for purpose objects with which it interoperates similarity with other objects constraints on object importance of object within use case adaptation of object to specific needs	functionality
failures in objects types of failures action which leads to failure of object	Reliability
naming of object mnemonic labels to objects understandability of object names understandability of object function frequency of usage of object layout/shape of objects handling of object presentation of object (interface)	Usability
time needed to operate resources needed to operate amount of data processed	Efficiency

Figure 62: Qualitative Aspects Related to Objects

ASPECT RELATED TO **ACTORS**	QUALITY CHARACTERISTIC
objects handled by actors interaction with system functions (which, how frequently) background of actors (educational, computer literacy, age ...) effect of background on task performance effect of background on quality of data output role of actor actions performed by actor interaction with other actors data handled by actors output produced by actors input produced by actors typical sequences of related transactions performed by actor adaptation of system to type of actor	functionality
effect of background on understanding of functions effect of background on handling functions effect of background on learning to use functions	usability
effect of background on quantity of data output effect of background on time needed for an action	efficiency

Figure 63: Qualitative Aspects Related to Actors

ASPECT RELATED TO **USE CASES**	QUALITY CHARACTERISTIC
typical sequences of related transactions variance in transactions objects handled during use case adequacy of processes for use case adequacy of data for use case adequacy of objects use case quality of output of use case actions involved in use case interaction between actors and objects in use cases processes involved in use case data accessed during use case customisation of data used during use case customisation of actions customisation of objects	functionality
help needed during use case understanding of actions involved in use case handling of objects involved in use case understanding of object names involved in use case effort to learn to use the system for use case	usability
stopping of actions during use case recovering data lost during use cases failures during use cases	reliability
time needed to perform use case quantity of output of use case resources needed during use case	efficiency

Figure 64: Qualitative Aspects Related to Use Cases

ASPECT RELATED TO **DATAFLOW DIAGRAMS**		QUALITY CHARACTERISTIC
flows	nature of flow nature of data transmitted (objective/subjective/manipulable...) size of data transmitted form of data transmitted where does it come from, where does it go correctness of data transmission (nothing lost)	functionality
stores	nature of data stored form of data transferred into/out of store origin of data in store (manual/automatic) quality of data retrieved from store procedure of building up data stores structure of data store interaction with other internal stores interaction with other external stores compliance with internal/external standards access rights to stores handling of data consistency in stores modifying stores: who, how customisability of data in stores	
processes	what is it for? what happens inside? data handled in processes nature of input into process nature of the transformation of input into output complexity of process (number and types of inputs, algorithms/procedures) quality of output of process characteristics of process input target characteristics of process output effect of background of actors to quality of output	
terminators	type of terminator (human/computer system) nature of terminator role of terminator within the diagram characteristics of terminator effect of characteristics of terminator (e.g. personal background) on processes effect of background on quality of data output data transmitted from terminator to processes data transmitted from terminator to stores data received by terminator from processes data received by terminator from stores role of terminator in external environment what happens with the data received?	
flows	understandability of information flow effort of transferring data effort of learning how to transfer data	usability
stores	effort of building up data stores effort of accessing data in stores (steps) effort of modifying data in stores effort of learning how to work with stores effort of understanding the structure of the store	
processes	effort of understanding the process	

ASPECT RELATED TO **DATAFLOW DIAGRAMS**		QUALITY CHARACTERISTIC
	handling of process inputs/outputs effort of learning to handle process inputs/outputs	
terminators	effect of background of terminator on: effort of understanding role of terminator in context effort of learning to handle flows/stores/processes effort of handling flows/stores/processes	
flows	interrupting flows: what happens to the data transmitted? possibilities to undo information flows	reliability
stores	reliability of data in stores (check routines etc.) recoverability of data in stores	
processes	failures occurring during process stopping of initiated processes undoing processes cancel running processes recovering data confirmation of critical processes escape functions	
terminators	reaction on errors on side of terminators error messages	
flows	time needed to transmit data resources needed to transmit data	efficiency
stores	volume of data stored limits to data stores resources needed to store data time needed to retrieve data from store	
processes	time needed to process resources needed to process effect of hardware/software environment on performance of process effect of background of actors on performance of process	
terminators	effect of background on quantity of data output effect of background on time needed for an action	

Figure 65: Qualitative Aspects Related to Dataflow Diagrams

ASPECT RELATED TO **DATA**	QUALITY CHARACTERISTIC
characteristics of data (char, boolean, integer, real) type of data (e.g. in NLP domain, text type) form of data (e.g. character sets etc.) complexity of data (language pairs, bi-multilingual) size of data in flows (strings, texts, etc.) size of data in stores correctness of data suitability of data for purpose availability of data to other processes availability of data to other actors availability of data to external systems integration external data security in access of data customisability of data compliance of data with standards	functionality
understandability of data (e.g. definitions) effort of handling data effort of learning how to handle data	usability
recovering lost data check routines before entering data consistency management of data	reliability
quantity of data processed in given time resources needed to store data resources needed to retrieve data	efficiency

Figure 66: Qualitative Aspects Related to Data

These lists are meant as a starting point that can in integrate findings of future evaluation processes. How the process of developing metrics can be guided by these lists of aspects will be demonstrated by the aid of the example below. The table shows how metrics for the *functionality* of the *data* in a termbank can be developed from the aspects given in the above figure.

QUALITATIVE ASPECT	METRIC	SCALE
characteristics of data	integration of graphics?	binary
type of data	information categories?	binary nominal
form of data	character sets?	binary nominal
complexity of data	languages? structure (concept/term oriented)?	binary nominal binary
size of data in flows	number of characters possible per term?	ratio
size of data in stores	limit in number of entries per language?	ratio
availability of data to other processes	integration of termbank into TM retrieval? access of termbank from editor?	binary
availability of data to other actors	multi-user access?	binary
availability of data to external systems	remote login? terminology export?	binary binary
integration of external data	terminology import?	binary
security in access of data	password check? different rights to different users?	binary binary
customisability of data	integration of new information categories?	binary
compliance of data with standards	terminology exchange formats?	binary nominal
suitability of data for purpose	adequacy of definitions? translations? context?	ordinal

Figure 67: Example for Using Qualitative Aspects of Modelling Concepts for Defining Metrics for a Termbank

When evaluating a translation memory system, the same qualitative aspects will inevitably lead to the definition of slightly different metrics that are appropriate for evaluating TM systems. Consequently, being generic in nature, the lists of qualitative aspects for the different modelling concepts satisfy the often recommended need for *reusability* of resources.

Once metrics are developed for those attributes relevant to the specific evaluation process, it has to be decided how the outcome of the tests will effect the adequacy of the software for the particular user, or in other words, how it will reflect decision makers preferences. Consequently, value functions for each metric under testing can be used in evaluation to express the extent to which the result fulfils the desired properties of the system, and at the same time, make the results of test of different systems comparable. In the software engineering context this concept is covered under the term “target value”, which describes what the ideal outcome of a test is.

3.2 Evaluation Preparation

The procedures used for modelling in evaluation strongly differ with respect to the type of evaluation situation. For evaluation preceding purchase decisions the aim of modelling is to identify *evaluation* and *value relevant* attributes while at the same time trying to keep the effort as low as possible. For evaluation supporting development, the aim of modelling is to identify any possible way in which the developer can benefit from the presentation of detailed user requirements. Consequently modelling in the latter case involves a great deal of effort and should deliver detailed information that needs to be integrated into the software development phase.

3.2.1 Preparing for Evaluation Preceding Purchase Decisions

Evaluation preceding purchase decisions ask for quality and utility assessment in numerical terms. The evaluation and assessment procedure that will be advocated in this thesis is one based on the definition of tasks as central element. The following steps can be applied as evaluation preparation:

1. Weighing the importance of the individual tasks to the evaluation process;
2. Elaborating task information, defining generic actions performed on specific objects; considering quality characteristics for generic actions and objects relevant to each task, and elaborating metrics measuring value relevant attributes;
3. Developing value functions for metrics;
4. Determining the appropriate test type for each metric.

An example will demonstrate the above depicted procedure. The example is based on the task description (cf. figure 28) as outcome of the domain featurisation process.

3.2.1.1 Weighting of Tasks in Domain

Evaluation preparation is a very time-consuming business. Effort should be directed mainly towards those tasks that are central in the domain. Consequently, in a first step, the evaluator has to discuss with the client of the evaluation process, how important the different tasks are. Here again it is important to note that different *stakeholders* may have different views about the importance of tasks. For user-oriented evaluation it is crucial to consider the views of users separately from those of decision makers. Whenever there is a clash between what decision makers and users consider important, a discussion between the two groups will have to resolve the problem. The following figure shows the outcome of the weighting process in the example.

task names	task short	weight users	weight decision maker	weight consolidated
administration	t1	.05	.20	.15
technical support	t2	.10	.15	.15
translation preparation	t3	.25	.20	.20
operative translation	t4	.60	.45	.50
SUM		1.00	1.00	1.00

Figure 68: Task Weighting in Evaluation Preceding Purchase Decision

The above figure shows that there is a big clash between the weights distributed by users and decision makers mainly with respect to the tasks *administration* and *operative translation*. The users in the example being translators in a translation department, naturally rate operative translation very high and administration rather low, mirroring the amount of time spent for the two tasks in every-day working life. In contrast to this, the decision maker can estimate an increase in overall productivity, if the effort in *administration* and *technical support* is reduced by the introduction of software. After consolidation, the importance of the *operative translation* task is still overwhelming, while also considering those aspects to a certain extent that make up the working environment of translation. It follows that the distribution of weights as demonstrated above is not only a precondition to the final utility calculation, but it also shows, where the major effort in elaboration of metrics has to be put.

3.2.1.2 Elaboration of Metrics from Tasks

In the context of evaluation preceding purchase decisions, the major objective of defining tasks in terms of *generic actions* and *objects* is to pinpoint those actions and objects that are of central importance and thus to reduce the effort of elaborating metrics and performing tests. The following table is based on the same task description and defines generic actions that are performed on specific objects during the *operative translation task*.

GENERIC ACTIONS	SPECIFIC OBJECTS
starting of objects	editor program translation memory program termbank program
opening of objects	SL text file TL text file translation memory data stores termbank data stores
initiating retrieval of objects	SL/TL sentence pairs from TM store terms from termbank
selecting best object	TL sentence TL term
initiating alternative search when no match is offered	parts of SL sentences terms using wildcards
browsing in objects for information	TM data store termbank
editing objects	TL text TM data store termbank
spell checking of edited objects	TL text TM data store termbank
storing of objects	TL text TM data store termbank
printing of objects	TL text
distributing objects	TL text updated TM data store updated termbank
exiting of objects	editor program translation memory program termbank program

Figure 69: Generic Actions performed on Specific Objects for Translation Task

In the *operative translation task* of the example we are concerned with 11 different *objects* and 12 different types of *actions*. At this stage it is already possible to rule out some of the objects that are not *evaluation relevant*. For instance, the evaluator can easily check with product documentation, which editor the systems are using. If each of the systems under evaluation makes use of WinWord, there is no need to elaborate metrics related to the *functionality, usability, reliability, and efficiency* of the object as such, only with respect to the interaction between the object and other objects of the list.

To illustrate the procedure of developing metrics from the qualitative aspects presented in section 3.1.3 the object *TM data store* will be considered in terms of *functionality* and the action *editing of objects* will be considered in terms of *usability*.

QUALITATIVE ASPECT	METRIC	SCALE
characteristics of object	language pairs? multi-directional?	binary nominal binary
type of object	database/files?	binary nominal
size of object	number of language pairs? number of translated pages? storage space needed? RAM space needed?	ratio ratio ratio ratio
operation modes	interactive/batch? translation segments: sentence/parts of sentence ...?	binary nominal binary nominal
appropriateness of object for purpose	match type: total/fuzzy suitability of fuzzy match proposals? success in variable handling?	binary nominal ordinal ratio
objects with which it interoperates	import of aligned segments? export of aligned segments?	binary binary
compliance with other objects	interchange formats?	nominal
constraints on object	attribute labels for TM databases: author/date/project etc.? type of variables defined: date/ names/...	 binary nominal binary nominal
adaptation of object to specific needs	changing of fuzzy match number? adding attribute labels for TM databases? definition of variables?	binary binary binary

Figure 70: Elaboration of Metrics for *functionality* of Object *TM Data Store*

Starting from the qualitative aspects presented for *actions* in 3.1.1, a great number of metrics could be defined that are relevant to the evaluation of a translation memory system. Many of the metrics developed above for TM data stores are also relevant to evaluating termbanks (e.g. language pairs, match types: total/fuzzy, import, export etc.). When considering other, less complex objects such as *SL/TL sentence pair*, the list of metrics will be much shorter, since only few of the qualitative aspects in section 3.1.3 can be applied.

The list of generic *actions* presented in figure 71 is particularly useful when evaluating the *usability* of a system. It shows that many of the actions performed by the user are rather similar, and therefore, the interaction with the system should be performed similarly. The following table presents metrics developed to measure the *usability* of a system with respect to the *editing of objects*.

QUALITATIVE ASPECT	METRIC	SCALE
effort to perform action	number of steps: (mouseclicks/keystrokes)?	ratio
different ways to perform action	type of user profiles? number of user profiles? user-definable shortcuts?	nominal ratio binary
difficulties in understanding actions	frequency of help usage for beginners? time spent with help?	ratio ratio
difficulties in performing actions	frequency of user errors? time spent to get back to work?	ratio ratio
compliance of interface in similar actions	adhering to interface standards? same steps used for actions that are subsumed under same actions type?	binary binary

Figure 71: Elaboration of Metrics for *usability* of Action *Editing Objects*

As the above tables show, the development of metrics for evaluation purposes involves a great deal of effort. For each evaluation environment it has to be determined, how exhaustively the development of metrics should be performed. In many cases, the evaluation budget will automatically lead to a selective development of metrics, that is, only for particular tasks, or for specific *objects* or specific *actions*.

With respect to the scales used for measuring, it is important to note that metrics where ratio, binary and nominal scales are applied, produce few problems in terms of *measurability* or *value relevance,* and should therefore be used whenever possible. In terms of *objectivity* these scales also generally rate high (unless measuring subjective notions such as "like" or "dislike" on binary scales). It is in many cases useful to transform nominal scales into binary nominal scales, which allow easy measurement in form of checklists. Representing notions of preference, ordinal scales naturally rate lower in terms of *objectivity* than ratio, binary and nominal scales. The variability of results of metrics using ordinal scales (from individual to individual, from occasion to occasion) has to be taken into consideration when determining the *reliability* of the metric. When interpreting the result of metrics using ordinal scales, the evaluator has to be careful with reading too much into the numerical representations of ordinal values. Similarly to decision analysis, user-oriented evaluation, however, cannot do without measurements on ordinal scales altogether. Ordinal scales always have to be critically examined in terms of their *value relevance*. An awareness of the dangers behind these scales helps to reduce the number of problems that may occur when making use of ordinal scales. The *validity* of metrics, finally, is a very critical issue but cannot be easily determined here. In general, the less abstract the metric applied in an experiment, the less the danger of it not representing circumstances in real life.

3.2.1.3 Developing Value Functions for Metrics

As the above examples of metrics show, many of the metrics are measured on binary or binary nominal scales. Consequently the values concerned are either **0** for not available/met or **1** for available/met. The following example illustrates how the target value for binary and nominal binary scales can be defined:

METRIC	SCALE	TARGET
changing fuzzy match number?	binary	1
definition of variables	binary	1
language pairs	binary nominal	
German - English		1
German - French		0
German - Spanish		1
German - Italian		1

Figure 72: Target Values for Binary and Binary Nominal Scales

Ordinal scales represent an ordered set of values, where value 1<2<n. Though ordinal value scales tend to produce monotone value functions, when it comes to utility, one may imagine that there may also be preference thresholds. The metric *suitability of fuzzy match proposals* illustrates this, where the proposals can be rated as:

0 not suitable at all
1 single words can be used
2 parts of sentences can be used
3 minor alterations necessary
4 variables of fuzzy match are automatically changed by system

The evaluation client may now say that, as long as only single words can be used, the fuzzy match proposal is useless. This will result in the following value function:

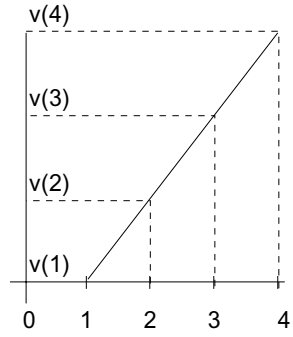


Figure 73: Example for Value Function with Threshold

Based on the above value function for the metric *suitability of fuzzy match proposals* the values of the options can be located on the curve as
v(0) = 0; v(1) = 0; v (2) = 32,5; v (3) = 65; v(4) = 100.

When developing value functions for ratio scales the evaluator has to define the acceptance level, that is, what is the upper limit of the scale. The next step is to consider whether there are preference thresholds, cut-off points or peaks. If this is not the case the function is likely to be *monotone.* Then one has to decide, whether the function is increasing or decreasing and which shape the function has. The following example illustrates this procedure: Considering the metric *frequency of user errors*, the evaluator decides the upper limit of errors per action should be 5, that is, if more than 5 errors occur during the performance of the particular action the value is 0. Furthermore the evaluator finds that "more is always worse than less" and thus concludes that the function is *monotonically decreasing*. To the client each additional error is equally important. In other words, the value decreases *proportionally* to the scale value, leading to a *linear decreasing function*. The following value function illustrates this:

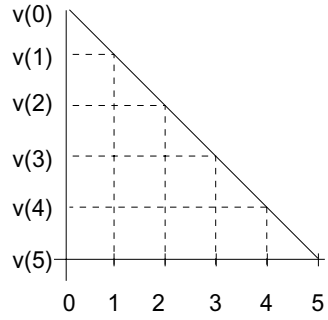


Figure 74: Example for Linear Value Function for Ratio Scale

Based on the above value function for the metric *frequency of user errors,* the values of the options can be located on the curve as
v(0) = 100; v(1) = 80; v(2) = 60; v(3) = 40; v(4) = 20; v(5) = 0

The elaboration of value functions as demonstrated above will inevitably lead to a number in the range between 0 and 100 for each metric applied to each system.

3.2.1.4 Developing a Test Model

In user-oriented evaluation, the development of metrics should be driven by the goals of evaluation, that is, what exactly do we want to find out about a piece of software; and related to this, the technique of testing, that is, how do we want to find it out.

The author of this thesis developed a model of test types during 5 years of practical evaluation work in the ESPRIT Translator's Workbench projects. It distinguishes between three test types with different goals behind testing, which will be described in

more detail in chapter 4. The following table only provides a brief overview which is relevant in the context of evaluation preparation.

TEST TYPE	SUB-TYPES	GOAL
scenario testing	field tests laboratory tests	to assess the appropriateness of a piece of software for every-day work
systematic testing	task-oriented testing interface-oriented testing benchmark testing	to examine the behaviour of software under specific conditions
feature inspection	---	to check the functionality of the software

Figure 75: Overview of User-Oriented Model of Test Types

In evaluation preceding purchase decisions it is of utmost importance to reduce the effort of testing as much as possible. Therefore, the first step should be, for each task under evaluation, to compile a list of all metrics that are relevant to the evaluation procedure and perform *feature inspection* by means of going through the documentation of the systems under evaluation. Binary or binary nominal scales are typical scales in feature inspection. There are, however, also a number of ratio scales that can be applied in *feature inspection*, such as for *number of termbank entries*, or *RAM space needed* etc. Many of the metrics applied during *feature inspection* will prove not to be *evaluation relevant*, since all the systems under evaluation have the same attributes, for instance *terminology import?* Y/N *terminology export? Y/N*. This does not, however, mean that these metrics are irrelevant, it only implies that they will not occur in the final assessment calculation.

For all those metrics for which values cannot not be obtained through feature inspection, the evaluator has to consider which test to use. For each of the remaining test types, that is task-oriented, interface- oriented, benchmark and scenario testing, a list of metrics has to be compiled prior to testing.

Considering the metrics for *functionality* and *usability* in the above example, derived from *objects* and *actions* involved in the sample task, the following distribution of test types is appropriate:

METRIC	SCALE	TEST TYPE
user-definable shortcuts?	binary	feature inspection
language pairs?	binary nominal	
multi-directional?	binary	
number of language pairs?	ratio	
storage space needed?	ratio	
RAM space needed?	ratio	
database/files?	binary nominal	
interactive/batch?	binary nominal	
translation segments: sentence/parts of sentence ...?	binary nominal	
match type: total/fuzzy	binary nominal	
import of aligned segments?	binary	
export of aligned segments?	binary	
interchange formats?	nominal	
attribute labels for TM databases: author/date/project etc.?	binary nominal	
type of variables defined: date/ names/...	binary nominal	
changing of fuzzy match number?	binary	
adding attribute labels for TM databases?	binary	
definition of variables?	binary	
frequency of help usage for beginners?	ratio	scenario testing
time spent with help?	ratio	
frequency of user errors? time spent to get back to work?	ratio ratio	
number of steps: (mouseclicks/keystrokes)?	ratio	task-oriented testing
type of user profiles?	nominal	
number of user profiles?	ratio	
adhering to interface standards?	binary	
same steps used for actions that are subsumed under same actions type?	binary	

Figure 76: Distribution of Metrics over Test Types

With the above list of metrics for each task under evaluation related to the test type by means of which it is to be assessed, evaluation preparation is finished and testing can begin. How testing is performed in evaluation preceding purchase decisions will be discussed and demonstrated by means of exhaustive examples in chapter 4.

3.2.2 Preparing for Evaluation Supporting Development

It has been pointed out before that evaluation supporting development has to be exhaustive and productive rather than assessable in numerical terms. In other words, the procedures involved in evaluation should be input into the software development process from requirements definition to operation and maintenance. For modelling in

the context of evaluation supporting development, the following questions are relevant:

1. What do we want to achieve through evaluation (goal)?
2. What type of test is appropriate?
3. What type of modelling is adequate to develop metrics for different test types?

For evaluation supporting development it makes sense to consider the different modelling concepts discussed in the software engineering section 3.1.2.1. The following figure shows the central concepts of the three modelling approaches and their relevance for the elaboration of quality attributes and metrics for the particular test types.

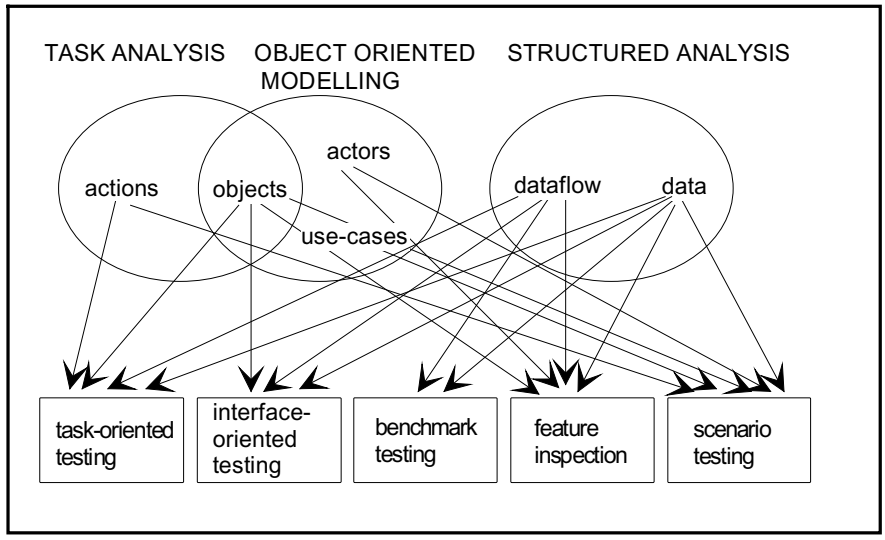


Figure 77: Relevance of Modelling Methods to Test Types

The modelling procedure in evaluation supporting development can be performed in two major steps, that is

1. Describe {D} and {M} by means of the modelling concepts that are most relevant to the specific test types;
2. Elaborate metrics along the qualitative aspects presented in section 3.1.3 considering the details elaborated in modelling {D} and {M}, and define targets.

3.2.2.1 Modelling for Scenario Testing

The above figure shows, if the motivation behind evaluation is to assess the appropriateness of a piece of software for every-day work, there are mainly five modelling concepts that are of interest, that is, *actions, objects, actors, use cases* and *data*.

Actions and *objects* are concepts stemming from Task Analysis (TA) and denote the activities performed by a person that are directed towards specific objects, where objects may be physical or informational. The generification of *actions* into generic task models as performed in TA provides an overview of what type of *actions* are performed with the system, in other words, it leads to the definition of standardised tasks. Moreover it is interesting to study the *frequency* with which a specific action is carried out during the performance of a task. Frequently performed actions require specific *usability* standards which need to be tested. In scenario testing the rough definition of *objects* provides merely a framework for the exact definition of actions. When modelling *objects* for the purpose of scenario testing, the most important activity is to *find* the central objects, while their organisation, interaction, operations and internal nature only need to be considered on a more general level. Thus there is no need to apply the complex procedures of object-oriented modelling for the modelling of all *objects*.

The modelling of *actors* and *use cases* is of central importance in the context of scenario testing. According to Object-Oriented Modelling *actors* represent the *roles* of the users (human or other systems). For scenario testing it is of utmost importance to define typical *roles* human users perform with the system, since depending on the *roles actors* play, different quality attributes will be relevant to testing. For instance, in the context of a translation department, a head of a translation group plays a different *role* than a translator. He/she will have different *actions* to perform, with different *objects* and different constraints on these objects (different read/write rights etc.). For scenario testing typical *use cases*, that is, typical sequences of related transactions performed by an *actor* and the system in a dialogue, have to be identified. Typical *roles* lead to the identification of typical *use cases*.

Finally the modelling of *data* is also important for scenario testing. It has to be found out, which type of *data* is involved during the performance of typical *use cases*. This is particularly important because a scenario test can only be performed successfully, when the relevant *data* is available and accessible during the test. In many cases this involves a careful and extensive preparation of the various types of *data* that are involved in the performance of a single *use case*. For scenario testing, it is sufficient to model the *data* on a very general level. It is based on the analysis of the *use case* selected for the test.

A scenario test is a test of one particular *use case*, where the *actions* of the subject are observed and checked against the performance of the system. During the scenario test

it will be examined whether and to which degree the system satisfies these attributes. Modelling *actions*, *objects*, *actors*, *use cases*, and *data* leads to the identification of attributes a system should have in order to support every-day work. These attributes need to be considered when preparing the instruments used during the tests such as scenario checklists or interviews. The following procedure for modelling requirements for scenario testing is most appropriate:

1. description: specific *actions* carried out by the user and specific *objects* handled by the user are described;
2. grouping: similar *actions* are grouped to establish subtasks and tasks;
3. generification: from specific subtasks and tasks more general task descriptions are elaborated, depicting standardised tasks;
4. definition: *actors* are identified;
   roles of *actors* are defined;
   use cases are defined for different roles;
5. analysis: *actions* performed in different *use cases* are analysed with respect to their frequency of occurrence;
   data involved in the performance of specific *use cases* is analysed along the following principle:
   1. Identify and name the *data* that is involved;
   2. Name the characteristics of the *data*;
   3. Define the relationship between the different *data*

3.2.2.2 Modelling for Systematic Testing

The principal goal of systematic testing is to examine the behaviour of software under specific conditions, covering three sub-goals and the related test types:

(i) examining whether the software supports the performance of pre-defined tasks: **task-oriented testing**
(ii) examining whether the functions offered work properly: **interface-oriented testing**
(iii) examining the performance of the system: **benchmark testing**

Different modelling concepts are relevant to the development of attributes that are tested in the different test types.

(i) Modelling for Task-Oriented Testing

In order to find out whether the software supports pre-defined tasks, it is necessary to look at *actions, objects, dataflow* and *data.* While scenario testing asks for the grouping of similar *actions* into generic task descriptions that lead to the description of

a standardised task, for task-oriented testing it is more interesting to study the *diversity* of actions that may be involved in the performance of a given task. In scenario testing the mere reason for modelling *objects* is to provide a framework for the description of actions. In task-oriented testing, however, *objects* that play a central role need to be considered in more detail. This is due to the fact that in task-oriented testing not standardised tasks or *use cases* are the central point of focus but rather specific object-action relationships. One way of describing *objects* was presented in the section on Object-Oriented Modelling, covering their interaction, composition, operations and nature. Another way that represents a certain description of the environment of *objects* that are related to a specific function was presented in the section on Structured Analysis, that is, *dataflow* diagrams.

The following procedure for modelling requirements for task-oriented testing is most appropriate:

1. description : *{D}:* specific *actions* carried out by the user and specific *objects* handled by the user are described;
   {M}: central functions are described in form of *dataflow* diagrams;
2. analysis: *actions* are analysed with respect to their diversity: which different possibilities are at hand to perform the task?
   objects and *processes* are analysed with respect to their importance: which are the critical *objects* and *processes* in the task?
   flows are analysed with respect to their nature: what type of data is transmitted?
   data and data *stores* are analysed along the following principle:
   1. identify and name the *data* that is involved;
   2. name the characteristics of the *data*;

(ii) Modelling Techniques for Interface-Oriented Testing

In order to examine whether the functions offered work properly it is necessary to look at *objects, dataflow* and *data.* The objective behind this type of test is **not** to find out whether the program suits a given purpose (like in scenario and task-oriented testing), but rather whether the given functions work properly, that is, without failure. To recall, in interface-oriented testing the software is examined from top to toe, considering each individual function as it is sequentially offered in the menu bar or in the windows. The best preparation for this type of testing is to gain an insight into the nature of the *objects* concerned and the *data* processed. *Dataflow* diagrams, which need to be developed for the central functions, form a basis for the pre-definition of the most central metrics.

The most appropriate procedure for modelling requirements for interface-oriented testing can be described as follows:

1. description: by means of learning what is relevant in both {D} and {M} the *objects* will be found (starting point: terminology of problem domain, software documentation); central functions are described in form of *dataflow* diagrams
2. analysis: central *objects* are analysed with respect to their
   interaction
   composition
   operations
   nature
   data is analysed along the following principle:
   1. identify and name the *data* that is involved;
   2. name the characteristics of the *data*;
   3. define the relationship between the different *data*

(iii) Modelling Techniques for Benchmark Testing

In order to examine the performance of language engineering systems the central concepts are *dataflow* and *data*. Benchmark tests can be applied either to individual functions, modules or to the overall system. To recall, in the strict technical sense, a benchmark test is the measurement of system performance without being dependent on personal variables. *Dataflow* diagrams in the first instance help to identify which functions or processes are suitable for benchmark testing, by investigating the nature of *terminators* and *flows*. Principally one may say that processes that communicate with non-human *terminators* fulfil this basic requirement of benchmark tests. In case of a human *terminator* the nature of the *flow* has to be defined. Only if the *data* transferred to a process is objective (e.g. Y/N), or not manipulable (e.g. entering a term in a terminology database) the process is suitable for benchmark testing. For the development of metrics and the selection of test data the nature of the *process* and the related *stores* have to be defined exhaustively.

Benchmark testing asks for the following procedure for requirements modelling:

1. description: central functions are described in form of *dataflow* diagrams
2. definition: for each dataflow diagram
   the nature of *terminators*
   the nature of *flows* and
   the nature of the *data* transferred is defined
3. analysis: for the function under testing

the nature of the *process* is specified
the nature of the *data* in the store is analysed along the following principle

1. identify and name the type of *data* involved
2. name the characteristics of the *data*
3. name the relationship between input/output *data*

3.2.2.3 Modelling for Feature Inspection

If the motivation behind evaluation is to check the overall functionality of the software, there are mainly four modelling concepts that are of interest, that is *actors*, *objects*, *dataflow* and *data*. Feature inspection aims at mapping the technical features of one or more systems onto the profile of one or more user groups or vice versa. Feature inspection is **not** concerned with the way functions are implemented but rather whether or not those functions that are considered important are present. Modelling for feature inspection, therefore, is concerned on the one hand with the rough description of systems in form of *dataflow* diagrams, and on the other with a description of typical roles of *actors*. The description of typical roles of *actors* usually precedes the description of the system. The interest of modelling roles of *actors* for feature inspection does not lie in the special sequence of transactions (*use cases*) like in scenario testing, but rather in the *objects*, which are involved in the execution of a specific role, that is, their interaction, composition, operations, and nature. The model of *objects* identified when describing the role of *actors* has to be mapped onto the *processes* and *terminators* of the *dataflow* diagram. The *data* handled by the users has to be mapped onto the data *stores* and data *flows*.

Feature inspection asks for the following modelling procedure:

1. description: *actors* are identified
   roles of *actors* are defined
   central functions are described in form of *dataflow* diagrams
2. analysis: *objects* related to the roles of *actors* are analysed with respect to their
   interaction
   composition
   operations
   nature
   data handled by *actors* is analysed along the following principle:
   1. identify and name the *data* that is involved;
   2. name the characteristics of the *data*;
   3. define the relationship between the different *data*

3. mapping: *objects* are mapped onto *processes* and *terminators* of the *dataflow* diagrams
data is mapped onto the data *flows* and *stores*

To conclude, it is important to note that being part of the development process, the primary aim of the evaluator is to provide as much *constructive input* into the development process as possible. This guarantees that the software will meet user requirements to a large extent. While in the context of evaluation preceding purchase decisions, the development of value functions is of great importance, at this stage, the evaluator supporting development rather has to set priorities with respect to the implementation of those aspects that are not yet properly considered.

3.2.2.4 Example for Modelling Process for Task-oriented Testing

Again, the following example will be based on the task description in figure 28. Details for *terminology look-up* as part of the *operative translation task* will be elaborated in order to show how modelling in evaluation supporting the development process is performed. The procedure follows the steps relevant to the process of modelling for task-oriented testing as described above.

The **description** step focuses on the gathering of the relevant information in order to gain an overview of the problem.

{D} The description of *actions* related to terminology in the operative translation task is part of the task description (starting of termbank, opening of termbank(s), accessing termbank from editor, searching terms, browsing in termbank, selecting termbank entries, pasting terms into text, editing terminology, updating termbank, entering new terms). The description of *objects* handled in context with terminology are termbank, SL term, TL term, terms with wildcards.

{M} Central functions of the operative translation task are demonstrated in the following dataflow diagram, modelling translator as *terminator*; segmentation, matching, term recognition, and term retrieval as *processes;* and SL text file, aligned SL/TL segments and term base as *stores*.

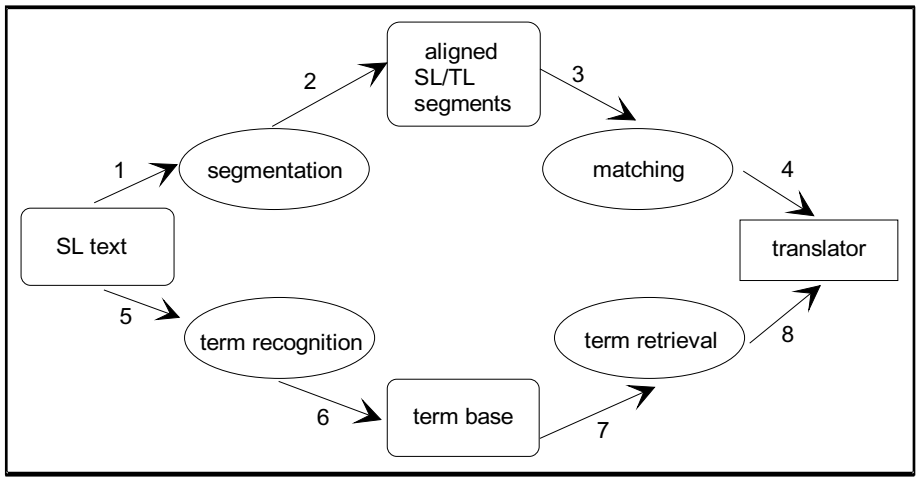


Figure 78: Dataflow Representation of Operative Translation Task

The *flows* in the above diagram transmit (1) SL text; (2) SL sentence or alternatively parts of SL sentence; (3) SL/TL sentence pairs; (4) TL sentence; (5) SL text; (6) SL terms or alternatively terms with wildcards; (7) SL/TL terms; (8) TL term.

Analysis involves the definition of the nature of those aspects described before. It leads to a clear understanding of the task, which is a precondition for the exhaustive development of metrics. As presented above, analysis asks for the consideration of the following problems:

Diversity of *actions* related to terminology look-up

Among the actions presented above, the critical ones are considered *searching terms* and *editing terminology*. The diversity of these actions will be further investigated:

terminology retrieval: are there different procedures for *searching terms*?

- automatic retrieval during translation: asks for term recognition;
- term lists prior to translation/interpreting: terms in specific text, or retrieval according to constraints (which ones?);
- typing in of terms during translation: possible problems w.r.t. character sets, maximal length of term, compound terms;

terminology editing and entering terms: are there different ways to edit terminology?

- editing during translation: asks for checking of rights of translator; handling of translation proposals of unauthorised translators?
- editing after finishing translation task: asks for way of storing information that needs to be added/deleted/changed.

Critical *objects* and *processes* in *terminology look-up*

term recognition: problems: lemmatisation, compounds, variations in compounds (e.g. man machine interface, human machine interface, user interface...).

term retrieval: problems: different hits (polysemes in SL); one hit with various options (multiple translations); no hit, which other strategies can be applied (wildcards etc.)

Nature of *flows* in *terminology look-up*

term flow: one "word" (character string from blank to blank) at a time: problems: number of characters limited? compounds, declined and conjugated forms.

Nature of *data stores* in *terminology look-up*

SL text: characteristics: language, text type, subject area, size;

termbank: characteristics: orientation (term, concept), languages, information categories, administrative info, size, entries, uni/multidirectional.

The above description and analysis of *terminology look-up* showed that the most critical actions are *searching terms* and *editing terms*. The starting point for the elaboration of metrics that measure the quality of the above two actions is the list of qualitative aspects relevant to the different modelling concepts (here: *actions*, *objects*, *dataflow* and *data*. The main objective in using the qualitative aspects presented in section 3.1.3 is to make sure that everything that might be of importance is considered. Naturally, when a number of different concepts are relevant, there is a certain overlap of qualitative aspects. For instance, the evaluator will find that the aspect of *interaction/interoperation* will be mentioned with regard to *actions*, *objects*, and *stores*. The best way to proceed, therefore, is

1. For each action under investigation, go through the list of aspects related to the different modelling concepts and pick out those that are important.
2. For each aspect relevant to the action, elaborate metric considering the problems identified in the analysis and description phase.

The following tables are the result of the process of selecting qualitative aspects denoting *functionality* and *reliability,* and elaborating metrics for *terminology retrieval* and *terminology editing*.

QUALITATIVE ASPECTS *functionality*	METRIC	SCALE
different options to perform action (search terms)	processes: automatic retrieval/ manual retrieval/ term lists?	binary nominal
suitability of output of automatic retrieval	number of terms in sentence/number of terms recognised lemmatisation component? languages supported with lemmatisation?	ratio binary nominal
suitability of output of manual retrieval	recognition of compound terms?	binary
suitability of output of term lists	terms in text/terms on list	ratio
constraints on retrieval	retrieval according to constraints possible?	binary
type of constraints on retrieval	type of constraints: author, subject area, date of entry ...	binary nominal
interaction with other actions	automatic retrieval integrated into output of TM retrieval?	binary
interaction with other objects	retrieval from editor possible?	binary
nature of data transmitted	usage of different character sets in editing window supported?	binary
size of data transmitted	max. number of characters for retrieval term?	ratio
nature of data stored	information categories?	binary nominal
origin of data in store	source of TL term available (author/date)? source of information categories available?	binary nominal binary nominal
structure of data store	term/concept oriented	binary nominal
interaction with other internal stores	access to different databases possible?	binary
compliance with internal/external standards	standard used: TIF ...	binary nominal
effect of characteristics of terminator (e.g. personal background) on processes	different user profiles for retrieval (translators, terminologists, interpreters....)	binary nominal
characteristics of data (char, boolean, integer, real)	integration of graphics?	binary
form of data (e.g. character sets etc.)	languages: en, de, fr, sp, ... which character sets are supported?	binary nominal nominal
complexity of data (language pairs, bi-multilingual)	unidirectional/multidirectional?	binary nominal
size of data in stores	number of entries per language? number of information categories?	ratio ratio
availability of data to other actors	multi-user? number of users possible?	binary ratio
suitability of data for purpose	suitability of definitions? suitability of contexts?	ordinal ordinal
QUALITATIVE ASPECTS *reliability*	METRIC	SCALE
failures during performance of actions	number of system breakdowns during retrieval?	ratio
types of failures	name failures and error messages	nominal
action which leads to failure of object	name steps that led to failure	nominal
possibility to stop actions	possible to stop retrieval process?	binary
escape functions	does escape stop retrieval process?	binary

Figure 79: Metrics for Task-oriented Testing of Terminology Retrieval

QUALITATIVE ASPECTS *functionality*	METRIC	SCALE
security in access of data	password check before editing?	binary
different options to perform action (editing terms)	options: editing during translation/ editing after translation/proposing changes by unauthorised translators?	binary nominal
procedure of building up data stores	"quick" editing possible (only minimal information)?	binary
type of input necessary to perform action	which info categories are a must for "quick" editing (minimal information)?	nominal
customisability of data	adding of new information categories possible?	binary
availability of data to other actors	edited info immediately available to other users?	binary
availability of data to other processes	stops system second editing process with same term by different user?	binary
consistency management of data	consistency/redundancy check during editing?	binary
interaction with external stores	copying of information from TM possible?	binary
origin of data in store	author/time label on each info category edited?	binary
size of data in store	limit of number or characters for info fields?	binary
suitability of edited data	check routines for entered info	binary
constraints on data objects	declaration of status of edited information (e.g. red/amber green) possible?	binary
QUALITATIVE ASPECTS *reliability*	METRIC	SCALE
confirmation of critical processes	confirmation window after editing? after each info cat? after each term? when changing to other processes?	binary nominal
possibilities to undo actions	after confirmation and saving, undo possible?	binary
possibility to stop saving process	does "escape" work while saving changes? other possibilities to stop saving process?	binary nominal
interrupting saving process: what happens to the data transmitted?	all changes preserved for new editing (going back to the point directly before hitting save)? or dismissing all changes?	binary
failures during editing process	number of failures during editing	ratio
types of failures	name type of failure	nominal
recoverability of data in stores	after system breakdown, newly edited info available?	binary
reaction on errors on side of terminators	error messages for user errors during editing?	binary

Figure 80: Metrics for Task-oriented Testing of Terminology Editing

Considering that the above metrics for task-oriented testing refer to only **two** actions in a translation environment as complex as the one described in the task description, it becomes clear, how much effort has to invested in the development of metrics for all tasks involved in the overall translation process.

4. User-oriented Testing for Evaluation

The test model, which was briefly mentioned in the previous chapter, will play a central role in the user-oriented evaluation of translators' aids' systems. It is based on general software engineering principles for software testing, while at the same time considering the specific problems in user-oriented evaluation. This chapter will outline the basic testing principles from software engineering in section 4.1. Section 4.2 will provide a detailed insight into the test model developed for evaluation. In section 4.3 the problem of test data generation will be discussed and examples given for the evaluation of translators' aids' systems. Section 4.4, finally, will present practical experiences and results of the user-oriented tests performed with two commercially available translators' aids' systems.

4.1 Testing Approaches from Software Engineering

Software testing is a rather complex area in which there is little consensus with respect to terminology or principal classifications. The US Institute of Electronic/Electrical Engineering (IEEE) is one of the sources for computing scientist that has produced a number of documents for guiding professionals in software testing and validation (IEEE 1059). However, IEEE is primarily aimed at software engineers rather than users. According to Sommerville (1996:pp.448) the testing process includes three major activities:

- **component testing** performed by programmers with individual units and modules after completion. It ensures that the program logic is complete and correct;
- **integration testing** includes tests on the sub-system and system level. The aim of the system test is to compare the system to its operational requirements and original objectives. It also validates system and product structure design;
- **user testing** is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the end-user rather than simulated test data. It aims at comparing the program to its initial requirements and current needs.

Of the above processes it is user testing that is to some extent directly applicable to the evaluation of translators' aids. However, it may also be possible to think of tests in evaluation supporting development, in which only individual components of the software are tested with users, for instance, only assessing the type of information categories offered in the termbank (translation, definition, context, grammar etc.). The concept of integration testing may also be relevant to tests in the evaluation of translators' aids, since it makes sure that the communication between the different

modules relevant in a translators' aids' system (editor, termbank, translation memory) works properly.

Principally, there are two different approaches to testing, that is, **glass box** (or white box) and **black box** testing, often also often referred to as code testing vs. acceptance testing, or structural vs. functional testing. In glass box testing, test design takes into account the knowledge of program internals. In black box testing, test design relies mainly on the knowledge of the system requirements.

4.1.1 Glass Box Testing Techniques Relevant to User-Oriented Evaluation

There is one manually performed glass box testing technique that is particularly relevant in the context of user-oriented evaluation, that is, program **inspection** (Fagan, 1976). Software inspection is defined in software testing textbooks as a group review process that is used to detect and correct defects in a software workproduct. It is a formal, technical activity that is performed by the workproduct author and a small peer group on a limited amount of material and produces a formal, quantified report on the resources expended and the results achieved. During inspection either the program or the design of a workproduct is compared to a set of pre-established inspection rules. Inspection processes are typically performed along checklists, which cover typical aspects of software behaviour, and involves examining by reading, explaining, getting explanations and understanding of system descriptions, software specifications and programs. While most testing techniques are designed to test one specific software quality characteristic, a major advantage of inspection processes is that any kind of problem can be detected and thus results can be delivered with respect to every software quality factor.

Software inspection as performed by software engineers may be adapted to user-oriented evaluation. While in glass box testing, the software engineer compares the source code to the specification along pre-defined rules and checklists, it may be possible, to compare a translators' aids' system in user-oriented testing to the requirements checklist by examining, reading and understanding both software system and user documentation. The fact that results can be delivered with respect to any software quality factor is another advantage that may be exploited in user-oriented testing.

Another glass box testing technique that may have some influence on the development of user-oriented testing methods is described by Sommerville (1996:471) and others as **path** or **branch** testing. It involves the execution of the program during which as many as possible logical paths of a program are exercised and requires that tests be

constructed in a way that every branch in a program is traversed at least once. Problems when running the branches lead to the probability of later program defects. The major quality attribute measured by path testing is program *complexity*.

It is the principal idea of path testing, that is, to follow each possible branch of the program at least once, that may be taken up in user-oriented testing. Even in user-oriented testing, it has to be made sure that the execution of each function offered by the translators' aids' system will not cause problems. Therefore, it is likely that similar tests in user-oriented testing will mainly measure the ISO quality attribute *stability* rather than *complexity*.

4.1.2 Black Box Testing Techniques Relevant to User-oriented Evaluation

According to Sommerville (1996:466) black box testing implies that the selection of test data as well as the interpretation of test results is performed on the basis of the functional properties of the software (sub) system. Among the most important black box tests are functionality testing, volume tests, stress tests, recovery testing, and benchmarks. Crucial for black box testing techniques is the identification of the mapping between 'inputs causing anomalous behaviour' by the system onto 'outputs which reveal the presence of defects'. This is of considerable importance for language engineering in that input is invariably vulnerable to ambiguity and the same is true for output. In textbooks it is repeatedly noted that the above types of black box testing should, if possible, **not** be performed by the author of the program. In new testing approaches, after the software developers successfully finished glass box testing, the testing of software systems is outsourced.

Functionality testing can be performed in different ways, either testing each program feature or function in sequence, or testing module by module, that is, each function where it is called first.

The objective of **volume tests** is to find the limitations of the software by processing a huge amount of data. A volume test can uncover problems that are related to the performance of a system. It uncovers aspects such as incorrect buffer sizes, a consumption of too much memory space, or, last but not least, response time problems.

During a **stress test** the system has to process a huge amount of data or perform many function calls within a short span of time. A typical example could be to perform the same function from all workstations connected in a LAN within a short period of time .

The aim of **recovery testing** is to make sure to which extent data can be recovered after a system breakdown. Does the system provide possibilities to recover all of the data or part of it? How much can be recovered and how? Is the recovered data still

correct and consistent? Particularly for software that needs high *reliability* standards, recovery testing is very important.

The concept of **benchmark tests** involves the testing of program *efficiency*. The efficiency of a piece of software strongly depends on the hardware environment and therefore benchmark tests always consider the soft/hardware combination. Whereas for most software engineers benchmark tests are concerned with the quantitative measurement of specific operations, some also consider user tests that compare the efficiency of different software systems as benchmark tests. In the context of this document, however, benchmark tests only denote operations that are independent of personal variables.

Each of the above listed black box testing techniques does, to a certain extent, affect the development of test types for user-oriented evaluation. Testing each function in sequence will make sure that each function is tested at least once and thus guarantees that the software fulfils its intended function. Particularly those functions that process natural language have to be closely examined in terms of inputs and expected outputs. Volume tests are important for translators' aids, since the amount of data processed, for instance, by translation memories may be huge. Stress tests can make sure that a translation memory or terminology database can be accessed via the net by a large number of users simultaneously. Recovery testing may be of some relevance to translation memory databases, since it can show, whether the data entered during the translation session is uncorrupted after a system breakdown. Benchmark tests, finally, play an important role in the context of the evaluation of language engineering products, since they measure the efficiency of these systems, which, to a large extent depends on the quality of the system output, that is, on the correctness and appropriateness of natural language text.

Acceptance testing is also a type of black box testing, since the generation of test cases is performed on a purely functional basis, involving real data and real users. In 1979, Myers (1979:114) complained that computing industry has placed insufficient attention on studying and defining good human-factor testing considerations. This is still valid today. For tests involving users, methodological considerations are rare in SE literature. Rather one may find practical test reports that distinguish roughly between field and laboratory tests, for instance in test reports by Karat (1990); Crellin/Horn/Preece (1990); or Moll/Ulich (1988). In the following the most important aspects of these tests will be described briefly.

In **field tests** users are observed while using the software system at their normal working place. Apart from general *usability*-related aspects, field tests are particularly useful for assessing the *interoperability* of the software system, that is,

how the technical integration of the system works. Moreover, field tests are the only real means to elucidate problems of the organisational integration of the software system into existing procedures. Particularly in the language engineering environment this problem has frequently been underestimated. A typical example for the organisational problem of implementing a translation memory is the language service of a big automobile manufacturer, where the major implementation problem is not the technical environment, but the fact that many clients still submit their orders as print-out, that neither source texts nor target texts are properly organised and stored and, last but not least, individual translators are not too motivated to change their working habits.

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

4.2 The Test Model for User-oriented Evaluation

Testing is the process of applying metrics and delivering attribute/value pairs. Considering both methodological attempts to software evaluation and practical user test reports in the broad software engineering area, one may roughly distinguish between three principal goals in user-oriented testing, that is,

(i) to assess the appropriateness of a piece of software for every-day work
(ii) to examine the behaviour of software under specific conditions
(iii) to check the actual functionality of a piece of software

During the TWB projects a goal-oriented model of test types was developed and proved to be appropriate for practical user-oriented evaluation problems (Höge/Hohmann/Le-Hong, 1995; Höge/Hohmann/Le-Hong, 1993; Höge/Kroupa, 1991). Following the above goals of testing, the test types are (i) scenario testing; (ii) systematic testing; and (iii) feature inspection. Special characteristics of the different test types in terms of testing environment; tasks; systems under testing; users; instruments, evaluation setup; and costs will be discussed. Moreover it will be investigated, which type of quality characteristics can typically be assessed by which type of test.

4.2.1 Scenario Testing

Though according to Karat (1990:352) the need to test systems in real work environments is receiving increased attention, there has been hardly any methodological attempt to define the exact nature of these kinds of tests. Myers

(1979:119) complained that user tests were often not considered the responsibility of the development organisation but rather that of the customer or end user, who, however, normally does neither have sufficient time nor the necessary software testing skills to develop proper methodologies.

The term "scenario" has entered software engineering in the early 1990s. According to Lubars/Potts/Richter (1993:14) or Gough/Fodemski/Higgins/Ray (1995:pp10), scenarios are considered a more informative way of conveying information both during requirements definition and testing. A scenario is based on the description of a specific *use case*, that is, it covers a special sequence of related transactions performed by an actor and the system in dialogue. It is a more comprehensive concept than *use case* in that it also considers the environment and its parameters. A scenario test is a test case which aims at a realistic user background for the evaluation of software as it was defined and performed, for instance, in the TWB projects and later also adopted by the EAGLES evaluation group. It is an instance of black box testing where the major objective is to assess the suitability of a software product for every-day routines. In short it involves putting the system into its intended use by its envisaged type of user, performing a standardised task. Of the two types of acceptance tests described in software engineering, that is, field and laboratory test, the field test comes closest to the concept of scenario testing in the model of user-oriented test types. Both types of user tests involve different testing environments, tasks, requirements on test system, user participation, instruments, testing expertise, and time and money constraints.

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 evaluator putting down notes, taking times etc. Karat (1990:352) points out that from a psychological point of view, the field test is considered to be the least obtrusive test in that it involves basically the same physical and social environment factors as normal work does. Among the physical environment factors, which are still likely to influence the behaviour of the user, are the layout of the office space, crowding and noise level. The most important social environment factors are office atmosphere and the normal pace of work (people stopping by and requesting information etc.). However, despite the advantage of displaying the every-day physical and social environment factors, a certain variance in behaviour can result from the psychological effects of being observed while working.

The task to be performed by different users during the field test should be standardised so that there is a chance that every user will encounter the same kind of problems and will have to perform similar operations to succeed. However, even standardised tasks

will be tackled differently by different users. As pointed out in chapter 1, each person has a different epistemic knowledge base and will apply different heuristic strategies to solve problems. The test task should correspond to the model, which was developed during evaluation preparation. Ideally the overall test task fits well into the organisational routine of the user's every day work and was developed beforehand in consultation with a number of users of the same environment. An advantage of field tests as compared to laboratory tests is that the test task can include problems of data transfer between the test system and existing systems. To ease evaluation, the overall test task needs to be divided into sub-tasks and actions, each identifying an operational unit of performance. For each sub-task or action the metrics that are of interest should be defined beforehand, so that the evaluator(s)' attention is automatically focussed on particular aspects of performance. The procedure of doing so will be described in more detail in the following chapter.

The development and application of metrics related to the test task are complex. The metric *time-on-task*, for instance, has two pleas on objectivity:

(i) apply same task with and without software support

(ii) keep personal variables constant

In the context of translation, comprehensive tasks are likely to involve rather complex, individually varying, problem solving strategies, which makes it difficult to compare results of metrics such as *time-on-task*. Colgan/Brouwer-Janse (1990:255) report of the problem that it is impossible to fulfil both pleas when considering complex tasks. On the one hand, applying the same task with and without software support would mean that the same user is confronted with the same task twice. Consequently, in the first test round a translator would be encountered with more problems, for which he/she has to develop strategies than in the second case, when he/she can tap his epistemic knowledge base to retrieve solutions which have been elaborated before. A variation of the test task, on the other hand, will lead to a different type and amount of problems that have to be solved by the user and will thus blur the test results to some extent. A variation of users is no solution either, since it will involve a different epistemic knowledge base and different heuristic strategies. Consequently the metric *time-on-task* can only be applied on the level of rather small sub-tasks, which do not involve complex problem solving strategies. An example for such a small sub-task would be: *look up term in dictionary*; test case (1) paper dictionary; test case (2) on-line dictionary.

Closely related to the problem of the test task are the requirements on the system under testing. If the test task can be considered as part of the daily organisational routine, the software system under testing needs to be in a highly operable condition. Thus field

tests are most beneficial, if the systems under testing are at least ß-versions of products to be launched in the near future or off-the-shelf products. The more the system presupposes a deviation of the task from the normal routine, the less informative are the results of the field tests.

For both kinds of scenario tests it is important that a representative number of users participate in the tests. According to Oppermann (1988:12) there are a great number of personal variables involved that can have a decisive influence on the performance of the system, that is, in all cases computer literacy, motivation or day-time. For the more complex language engineering applications, such as translators' aids' systems, furthermore education, experience and expertise need to be considered. The organisational environment of field tests, which do not involve much extra expenditure for equipment etc. normally allow the participation of more subjects than in laboratory tests of comparable costs.

The instruments commonly used in field tests range from the simple observation of users and noting their behaviour on evaluation checklists, to pre-and post-testing interviews, think aloud, and, last but not least, logfile recording, each of which will be briefly discussed here.

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

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

According to Moll/Ulich (1988:73); or Crellin/Horn/Preece (1990:330) pre-testing interviews are performed in order to elicit the subjects' personal background, opinion and expectations concerning the system that is going to be tested. The information gained by means of pre-testing interviews gives valuable hints when interpreting the scenario test results. Testing in the TWB I and II projects proved that post-testing interviews are an important, if not necessary, part of each scenario test. They are performed after the observational data (video tapes or checklists) and logging protocols are analysed. Each aspect that needs further clarification is taken up in the post-testing interview. When performed in conjunction with video observation or think aloud, the behaviour and comments of the subjects in particular situations can be discussed with the subject and analysed jointly. A combination of both pre-and post-testing interview is particularly useful, since it also allows the assessment of the change of mind of subjects during the testing exercise. Moll/Ulich (1988:pp73), for instance, reported that at the beginning of a test, attitudes towards the usefulness of help systems were quite positive, while at the end, after having used the help system various times it was much more negative.

Think aloud protocols are used in many empirical investigations. They are a means of qualitative data collection. Vainio-Larsson (1990:325); Moll/Ulich (1988:74); and Crellin/Horn/Preece (1990:331) point out that the motivation behind using think aloud protocols is to collect information on the users' own reasons for their behaviour or, as Goguen/Linde (1993:156) put it, to get a direct verbalisation of specific cognitive processes. According to Cordingley (1989:143), instructions for the knowledge provider are likely to include:

- say out loud everything you are thinking from the first time you see the problem until you solved it
- talk aloud constantly
- do not think about what you are going to say
- do not explain what you are saying

Criticisms of think aloud protocols are based on doubts concerning their validity and reliability. Goguen/Linde (1993:157), for instance, argue that think aloud protocol analysis "... is based on a simplistic cognitivist model of human thinking as essentially computational, involving abstract representations of concepts, and their transformation by algorithms that are precisely specified by computer programs." Moreover, think aloud protocols presuppose that users are able to describe their actions, which is often not the case for routine processes. Think aloud protocols are only appropriate for subjects who are trained to verbalise their thoughts. Also, Hönig (1991:82) argues that

what users are able to verbalise, represents only the conscious part of their thoughts and thus neglects important subconscious aspects. Another problem of applying think aloud protocols is that it may have a negative effect on the user behaviour it is even more intrusive than pure observation, and, finally, Vainio-Larsson (1990:325) complain that "... many users have difficulty in acting and reflecting simultaneously." Due to criticism concerning both validity and reliability of think-aloud protocols, in testing practice they should only be used as a complementary method. As such they are valuable, since they can provide clues to problems that stem from the interpretation of data gathered by means of other techniques

Logging and playback programs are general data collection programs that can be used with actual product code or prototypes of the user interface of a product under development. Vainio-Larsson (1990:325) argue that recording not verbalised operations, that is, all keystrokes and mouse activities, including incorrect inputs, provides useful information on quality characteristics related to the *usability* and *functionality* of the software. For instance provides the frequency of use of a certain function within several testing sessions some hints on the *task-relevance* of the function, the occurrence of cumulative handling errors of users provide information on the *understandability* as well as on the *learnability* of the function, the *suitability* of the help function can be assessed from the number of cases in which after the consultation of help solutions were found etc.. Diaper (1989-3:229) points out that the major advantage of using logging and playback programs is that they work automatically, are error-free and broaden the scope of results, since they provide the evaluator with a large amount of extra data and insight. Contextual information on the user behaviour, however, which is vital for correct data interpretation, has to be elicited by means of additional instruments such as observation or interviews.

The choice of instruments depends on various factors such as time and money constraints, technical facilities, evaluation expertise etc. Due to the limited possibilities of retrospective data analysis present in field tests, it is important that the data gained with the aid of the different instruments (notes on user behaviour, interaction etc.) be analysed right after finishing the test, because otherwise important contextual information is likely to get lost.

The evaluation setup of field tests generally puts heavy demands on the expertise and experience of the evaluator. The system under testing needs to be organisationally and technically integrated into the existing environment. The normal working routine should be interrupted as little as possible during the test. Whereas laboratory tests provide the evaluator with various possibilities to record and re-play the different test

situations, evaluators in field tests mostly have to rely on what they identify as important information during the various situations in a test.

The final costs of a scenario depend on personnel and equipment. The major difference in costs between field and laboratory tests lies in the equipment. Field tests ask for comparatively little equipment expenses because hardly any investment in additional technical evaluation instruments is obligatory. Thus, according to Karat (1990:355) field tests mostly invoke less costs than their laboratory counterpart.

The following table outlines the major differences between field and laboratory tests in terms of those parameters that were discussed above.

	FIELD TEST	LABORATORY TEST
testing environment	normal working place least (but still slightly) obtrusive same physical/social environment factors	laboratory obtrusive new working environment integration of developers into tests possible
test task	representative integrated tasks fits into every-day routine includes problems of data transfer	non-integrated tasks possible to test specific modules only
test system required	operable system or ß version	prototypes or operable systems
users	more users/budget	less users/budget
instruments	direct observation think aloud checklisting pre- and/or post-testing interviews logging programs	indirect observation - one-way mirrors - video recording - audio recording think aloud logging programs
evaluation setup	technical and organisational integration into existing environment interruption of working routine	experimental setup no integration into environment
comparison of costs	moderate	high

Figure 81: Field Test - Laboratory Test - A Comparison

Of all test types it is mainly the scenario test that can provide the most detailed information on the quality subcharacteristics *understandability*, *learnability* and *operability.* Additionally scenario tests can provide detailed information on *suitability*, *interoperability* and *customisability*. Depending on the test task, information can also be elicited on a system's *maturity*, *fault tolerance*, and *recoverability*, as well as on *time-* and *resource behaviour.* In addition to these central characteristics, the problems encountered during system installation and adaptation may provide information about a system's *changeability* as well as about *installability*, and *adaptability.* Karat (1990:353); Lewis/Henry/Mack (1990:338) and many more report that typical metrics applied in scenario tests are *time on task*, *completion rate*, *error free rate*, *time needed for training programme, frequency of help/documentation use etc.*

The quasi experimental design in scenario tests delivers to a large extent subjective results. The most commonly used techniques to reduce subjectivity in scenario tests, therefore, are to calculate averages and variances on a sufficiently large number of subjective judgements, while trying to avoid inferences with other systems. However, striving for a "cleanroom" approach for scenario tests by selecting test persons and subjects that do not to have inferences with other systems is dangerous, since while achieving more objectivity, the results are likely to become less representative. Thus, in user-oriented evaluation the conviction is shared with behavioural scientists like Edwards/Guttentag/Snapper (1975:145), who argue that it is often not possible to study user-related issues without making use of quasi or pseudo experimental design, despite all its shortcomings in terms of objectivity.

4.2.2 Systematic Testing

Under the term systematic testing all testing activities are 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 testing, systematic tests can be performed solely by software engineers and/or user representatives. Systematic testing follows three major objectives:

(i) examining whether the software offers functions for pre-defined tasks;
(ii) examining whether the functions offered work properly; and
(iii) examining the performance of the system functions.

Accordingly user-oriented systematic testing will be split up into task-oriented testing (section 4.2.2.1), interface-driven testing (section 4.2.2.2), and, benchmark testing (section 4.2.2.3).

4.2.2.1 Task-oriented Testing

Task-oriented testing is performed to examine whether and the extent to which a piece of software offers functions to perform specific tasks. Task-oriented testing is related to scenario testing in that its major purpose is to assess the overall functionality of the system by means of relevant data inputs, as well as to examine the quality of the data output. Yet, there are several reasons why one may decide in favour of task-oriented testing instead of performing scenario tests, that is,

(i) restricted budget
(ii) time constraints
(iii) no users available
(iv) no laboratory available
(v) functionality of prototypes restrictive in performance

The final success of task-oriented testing lies in the exhaustive definition of a representative number of test tasks and subtasks, which should be both relevant to the application domain, and supported by the system under consideration. It focuses on performing *critical* functions of a system, that is tasks that involve a complex interaction between *processes, stores*, and *terminators* on the one hand, or *objects* and *actions* on the other. Whereas in scenario testing, the test task needs to be standardised, task-oriented testing can define a broad range of test tasks, which may all be relevant to one user or the other. Whereas in field tests, for instance, problems in the performance of the test task (e.g. system failures etc.) lead to an interruption and therefore a rather costly failure of the whole testing process, task-oriented testing allows a repetition of test tasks, while documenting the problems encountered. While in scenario testing each sub-task has to be defined beforehand, task-oriented testing leaves more space concerning an investigation of the possible ways of performing a given task with a given system.

Testing instruments that are applied during task-oriented testing are mainly restricted to checklists containing the tasks, sub-tasks and related metrics and test problem reports. Test problem reports provide developers with the detailed description of problems that occur during testing. According to Thaller (1993:123) they can be very important instruments in the context of evaluation supporting development, since they aim at the improvement of the software. The most important part of test problem reports is the detailed description of the problem and the actions that led to the problem. A diagnosis of the failure is given and the action required is described, if possible. Another important aspect that needs to be noted in test problem reports is the priority ID of the failure. The following failure priority score, which is presented by Deutsch (1982:289), can be considered generally representative.

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

Figure 82: Failure Priority ID Score

A sample test problem report sheet of the TWB Projects will be provided the appendix 1.

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

The primary quality characteristic under investigation in task-oriented testing is *functionality*. In this sense task-oriented testing comes close to what software engineers call "functionality testing," that is, investigating whether the program does what it is supposed to do. Being able to test a great number of different tasks, the *suitability* of the software, that is, the *presence, appropriateness* and *accuracy* of a set of functions for specified *tasks* can be closely examined. When performed at the final installation place of the software, task-oriented testing can also deliver valuable results concerning the *interoperability* of the software. For this purpose, the tasks must cover the communication with other applications and/or users within the given environment. The *compliance* of a system's user interface, as well as system interface- and data formats to standards can be tested thoroughly. It can be tested whether the system meets required *security* standards for the access of data stores or the performance of functions. Also, the effect of the system's *customisability* on the output of the data can be examined. Task-oriented testing can further provide some information on the system's *reliability* in terms of its *maturity* (relating to the problems encountered during testing), its *fault tolerance* (related to the problems of incorrect inputs) and *recoverability* (related to incorrect actions). In addition, the *usability* of the system can be assessed - though with a different focus and in a different way than in scenario testing. One of the most frequently applied metrics of *operability* in task-oriented testing, for instance, is counting of steps necessary to perform a certain task, or evaluating the user interface layout (interface fonts, windows, icons, buttons etc.). Similar to scenario testing, there is no particular focus on testing *efficiency* during task-oriented testing. Nevertheless some results may be obtained concerning a the *time-* and *resource behaviour* of a system while performing specific tasks.

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. Users are only involved in the definition of the test task and not as subjects. Consequently, the overall organisation of task-oriented testing is less demanding than for scenario tests.

4.2.2.2 Interface-driven Testing

While both scenario and task-oriented testing are mainly geared to examine the handling and functionality of the software, the philosophy behind interface-driven

testing comes closest to the software engineer's principal aim, that is, the discovery of software problems. Interface-driven testing follows the software engineer's principle of testing by following each possible path or function in sequence. Consequently, while in both scenario and task-oriented testing only particular functions are performed, namely those that are necessary to perform the test tasks, in interface-driven testing each function of the software is executed at least once.

Similar to task-oriented testing, interface-driven testing can be performed at any stage of the software life-cycle as well as with off-the-shelf products. The test does not have to fit into any operational environment and does not involve users. Instead of following pre-defined tasks, the evaluator explores any possible way of handling the system. The modelling of central functions, objects and data is the basis for the definition of the most important metrics. However, as is also true for task-oriented testing, not every metric relevant to interface-driven testing can be defined prior to testing. This is due to the fact that the test is *not* performed along a certain pre-defined task and the evaluator, therefore, cannot anticipate at what stage he/she will encounter which type of function. It may be necessary, therefore, that the evaluator has to develop metrics and elaborate test data, perform tests and document the results on an ad hoc basis while executing the software. Only when executing a certain function, the evaluator can guess which data is needed to perform certain operations with the functions offered by the user interface (the WHAT HAPPENS IF ... test). The costs of interface-driven testing mainly lie in the recruitment of experienced evaluation personnel that is capable of the ad hoc generation of metrics and data.

In interface-driven testing the major focus lies on delivering results on the system's *reliability*, including characteristics such as *maturity*, *fault tolerance* or *recoverability*. In the case of a terminology elicitation system, for instance, a volume test can be performed by executing the option "do concordance" with an unusually big file. For a translation memory a stress test could be performed, accessing, for instance, a certain number of parallel texts by different users on the LAN at the same time. Recovery tests could be performed with a termbank when, e.g. simulating a system breakdown (e.g. on a PC by the key combination control/alt/del) before and after having properly saved terminology modifications. Apart from *reliability*, interface-driven testing delivers results about a system's *functionality*. Valuable results can be elicited concerning the system's *compliance* with other standards, e.g. whether a windows application consistently makes use of the same windows system messages as other applications do, or whether the system is internally consistent concerning its naming of functions and processes etc. Also system *security* is one of the characteristics that only interface-driven testing can sufficiently investigate, since unlike in task-oriented and

scenario testing, it is certain that every function is performed and checked. If performed in the environment where the system is going to work, the *interoperability* of the system with other devices can be assessed. Finally, the *customisability* of the overall system can be tested. The testing of *usability* is possible in interface-driven testing, though it is not a major focus. The results about *understandability* and *learnability* are subjective, going back to only one individual. The results on *operability* are not necessarily as subjective. The counting of steps to perform functions, for instance, is an important objective metric that can be performed for all functions and compared between systems. Interface-driven testing does not focus on testing the *efficiency* of a system. In practical terms, interface-driven testing often allows the identification of functions that do not involve personal variables, that is, which can at a later stage be benchmarked. In rare cases, results are obtained about *time*- and *resource behaviour*.

4.2.2.3 Benchmark Testing

The benchmark test examines the performance of systems, either of individual functions and modules, or of the overall system. In the strict technical sense, a benchmark test is the measurement of system performance without being dependent on personal variables. Thus, following the narrow definition of benchmark, there are very few possibilities to apply benchmarks on the module or even system level of interactive systems. Software engineers like Lewis/Henry/Mack (1990:337); or Oppermann (1988:12) use the term *benchmark* in the wider sense to denote the comparison of the overall performance of different interactive systems. In the context of this thesis, however, the term benchmark will only be used in its original technical sense, that is, denoting objective, reliable system measurement. Examples of benchmark tests in the language engineering area are on the function level, e.g. the measurement of success rates for automatic terminology retrieval functions, the measurement of translation retrieval rates for translation memories, or the measurement of time for the parsing of a text. For a detailed description of a number of benchmarks applied with translators' aids' systems see Schüler (1995).

A benchmark tests in the strict technical sense roughly involves the following activities:

- the identification of functions independent of users by means of modelling
- the definition and elaboration of data that the function is supposed to process
- the definition of the expected correct data output
- the investigation of possible sources of error
- the measurement of time or resources spent.

Benchmark tests allow the comparison of the performance of different systems. When performing the same benchmark with different systems, it has to be kept in mind that both system parameters and environment variables be kept constant. Only if different translation memories, for instance, have access to the very same background material and are tested with the very same test text, the comparison of the benchmark results makes sense.

The metrics applied in benchmark tests measure largely *time* and *resource behaviour.* For language engineering applications such as translators' aids there are also benchmarks measuring the *accuracy* of system output both in terms of their correctness and the expected quantity (e.g. recall rates). The type of results achieved by means of this type of benchmark tests are mostly numbers, e.g. the time needed to perform a certain function, the resources needed when performing a function, the amount of output data produced in a given time or the ratio between input and correct output data.

4.2.3 Feature Inspection

The aim of feature inspection is to describe the technical features of a piece of software as detailed as possible, so that it allows the comparison between systems of the same type. There is a conceptual similarity between feature inspection and the glass box concept of inspection, which involves the comparison between a piece of software and a pre-defined feature checklist. Similarly to glass-box inspection, it is essentially a manual testing technique that does not necessarily involve the execution of the program. To a certain degree, feature inspection can be performed on the basis of the software documentation. However, while its glass box counterpart is a means to actively seek for software problems, feature inspection in user-oriented evaluation has a more descriptive character, that is, checking the availability of features rather than the absence of errors.

Spies (1995:97–115) presents a detailed comparative feature checklist for translators' aids' systems. Successful feature inspection depends mainly on the quality of the feature checklist along which the evaluator examines the software. Feature checklists incorporate mainly binary nominal, in some cases also ratio scales. Particular attention has to be paid to the definition of user profiles, capturing *possibly disjoint* requirements of the customers of the evaluation process. In practical terms this means that the evaluator elaborating a feature checklist needs to study a broad range of user requirements including organisational constraints of *different* setups. Moreover, the evaluator has to get acquainted with a broad range of systems of the same type, in order to be able to grasp (possibly disjoint) underlying philosophies.

Any feature checklist in the context of evaluation needs to be both standardised in the sense that it should be independent of situational variables and open in the sense that it can cover different approaches to a problem without being prescriptive in nature. Since most feature checklists are based on the state of the art of development, they only describe features that are common technology. If, however, new technical solutions have been found to an "old" problem these solutions are not likely to be instantly part of feature checklists..

Similarly to glass box inspection, feature inspection can tackle every quality characteristic. The major focus, however, lies on investigating the system's *functionality*. As pointed out before, feature inspection is not concerned with the way features are implemented but rather whether or not those features that are considered important are present.

4.2.4 Conclusion to the Model of User-Oriented Test Types

The three test types discussed above are based on common software engineering principles and are the result of constant reconsideration and refinement on the basis of both practical test cycles in an industrial translation environment and scientific evaluation research. Practical testing experience showed that the three test types take into account the specific requirements of user-oriented evaluation of language engineering products more than any of the existing testing models. Existing glass and black box testing techniques and methodologies are geared to testing as part of the software life-cycle and the software engineer as agent of evaluation. The user-oriented model discussed above is geared to the testing of language engineering systems by evaluation agents who do not necessarily have a computational background.

However, as any model asks for categorisation and simplification, the model of test types developed in this section may also not be able to directly accommodate every situation imaginable in the context of evaluation. Moreover, for existing testing environments that were not primarily elaborated according to the above goal-oriented approach, it may be difficult to fit individual testing procedures into one or the other category of tests. Particularly complex setups always have asked and always will ask for hybrid test types. Thus the model has to be understood as point of orientation in the jungle of evaluation techniques rather than a fixed frame into which any setup has to be pressed.

4.3 Test Data Elaboration in User-Oriented Testing

In a technical sense, a test establishes a quality relationship between system inputs and outputs. The selection and preparation of test data, therefore, is closely linked to the test types and instruments, which are to be used in a test. IEEE 1059:15 defines the principal goal in the selection or elaboration of test data to be to uncover errors, omissions, and unexpected results. In user-oriented testing, however, the principal goal is to determine how well a system does what it is supposed to do and only secondly to uncover errors, omissions and unexpected results. Another difference between user-oriented and development-oriented testing that has to be noted in connection with test data selection is that in user-oriented testing the selection or elaboration of test data *cannot* be based on a deeper understanding of the program internals. What is it then that drives the selection of test data in user-oriented evaluation of language engineering products? In order to throw some light onto this question the following chapter strives to

- give a brief overview of general issues related to the selection or elaboration of test data in the context of software engineering as well as what is relevant to user-oriented evaluation;
- present the types of test data that are most frequently used in the language engineering context, their characteristics, advantages and drawbacks;
- elaborate which parameters have to be considered for test data selection and elaboration in user-oriented evaluation of language engineering products; and
- give an example for test data elaboration to evaluate the adequacy of a translation memory component for an international car manufacturer.

Balkan et al (1994-1:pp.27) identify three characteristics of test inputs (or test items as they are frequently called) that are of particular interest to NLP evaluation, that is, nature, coverage, and origin.

- The **nature** of test items for language engineering applications denotes both linguistic and extra-linguistic phenomena. Linguistic phenomena may cover instances of morphology, syntax, semantics, extra-linguistic phenomena include numbers, acronyms, formatting, punctuation, lists, figures etc.
- The **coverage** of test items refers to the degree to which different phenomena are used as test items (breadth of coverage) and to the degree to which a combination of phenomena is tested (depth of coverage). In the framework of TSNLP (Test Suites for Natural Language Processing), Balkan et al. (1994-2:6) give the following example of high level phenomena:

  breath of coverage: morphology
  syntax
  semantics

	extra-grammatical
depth of coverage:	ill-formed data
	interaction and co-occurrence.

- The **origin** of test items is of particular importance to the evaluation of language engineering systems. Particularly the question whether test items are artificially constructed to cover specific phenomena, or based on or even extracted from real text? A frequently applied distinction of test data with respect to its origin is between **test collections**, **test suites** and **test corpora**, which will be discussed under 4.3.2.

4.3.1 Approaches to Select System Inputs

In the software engineering context, there is a strong need for vigorously choosing test data that, on the one hand does not influence the development strategy, but on the other provides good feedback upon use to both developers and users. In general, data input for testing purposes can be characterised with respect to its **validity**, where valid data includes all inputs a program should be able to process and invalid (or erroneous) data includes all inputs that a program should not process, that is, should mark or reject. The definition of both valid and invalid inputs is based on the functionality of the program, that is, what exactly is the program supposed to do? A simple example would be a client database, which at some stage requires the client's date of birth as input. Offering six blank characters as input space, the only type of valid input is number: day/month/year. Instances of invalid input would be characters instead of numbers, or numbers over 31 for day, over 12 for month.

Myers (1979:36-46) discusses two approaches to test data selection, that is, **random testing**, where test data is selected or generated randomly and **functional testing**, where functional properties of the system guide the selection of test data.

The most important classical approach to the selection of test data in functional testing is *equivalence partitioning*. It is based on the fact that programs normally behave in a comparable way for all members of a class. It aims at selecting those subsets of possible system inputs with the highest probability or finding the most errors. This involves the identification of different input phenomena and the partitioning/categorisation of these phenomena into a finite number of equivalence classes such that one can reasonably assume that a test of a representative value of each class is equivalent to a test of any other value of that same class. The identification of equivalence classes is a heuristic process that starts from statements in the system specification. For each condition in the system specification, classes of inputs are specified that the system is supposed to process (*valid equivalence classes*)

and those that the system is supposed to mark (*invalid equivalence classes*). Myers, provides the following tabular example for the elaboration of equivalence classes for a simple condition stated in the specification of some program:

EXTERNAL CONDITION	VALID EQUIVALENCE CLASSES	INVALID EQUIVALENCE CLASSES
"the item count can be from 1 to 999"	1<item count<999	item count <1
		item count >999

Figure 83: Valid and Invalid Equivalence Classes from Myers (1979:46)

The second important classical approach to test data selection is *boundary value analysis*, in which the boundary conditions of a system are explored. According to Myers (1979:50), boundary conditions are those situations directly on, above, and beneath the edges of input equivalence classes and output equivalence classes. This mainly involves the selection of test inputs that represent the ends of accepted ranges of inputs, such as maximum and minimum values.

It is interesting to note here that apart from the above two generally acknowledged approaches to test data elaboration, Myers presents another approach which he calls *error guessing*. Given a particular program, evaluators surmise, both by intuition and experience, certain probable types of errors and then write test cases to expose these errors.

In the context of testing language engineering products the selection of test data is an important issue, since the complexity and ambiguity of language needs to be taken into account in order to deliver reliable results. A mixture of the above approaches from software engineering may be useful to select relevant test data in evaluation of translators' aids' systems. Of the three approaches discussed above *error guessing* is the one that comes closest to the approach that is naturally followed in user-oriented evaluation. It is based on a thorough understanding of the problem domain and a general insight into the capabilities of the systems under consideration. *Error guessing* delivers results on the behaviour of a system in those situations that are considered critical. However, only making use of *error guessing* in the elaboration of test data and test case design would lead to a very low coverage of test cases. Consequently *error guessing* needs to be complemented by engineering strategies that will lead to a higher coverage of test cases in user-oriented evaluation. The approaches to *equivalence partitioning* and *boundary value analysis* are particularly interesting in this respect. The identification of classes of inputs that share principal characteristics and therefore are likely to yield similar results is relevant to both the detection of errors and the definition of how well a system does what it is supposed to do. In user-oriented testing

of language engineering systems the identification of classes of input is impeded by the following problems:

(i) testing is performed on a higher level, that is, on the level of integrated systems rather than of modules;
(ii) in a black-box situation, the evaluator has no access to program internals;
(iii) the processing of natural language is not a totally rule-based problem.

Due to the above presented facts, in user-oriented evaluation of language engineering systems, the classical approaches cannot be performed exactly in the same way as in the development context. Yet what can be taken over is the way to approach the problem of data elaboration. The following strategy can be employed in user-oriented evaluation for each function to be tested:

1. **Problem domain definition**: identify roughly what the user expects of a particular function in terms of what type of data should be processed and how the results should be (specify attributes of {D}).
2. **System functionality definition**: identify roughly what the system is supposed to be capable of doing (specify attributes of {M}).
3. **Error guessing**: consider which situations are particularly error prone (experience/intuition).
4. **Test cases definition**: define the type of input for error-prone situations.
5. **Input text examination**: examine the type of data that should be processed in terms of its characteristics.
6. **Input categorisation**: organise the characteristics of the data into different categories that relate to the same problem.
7. **Test cases definition**: identify test cases for each category.
8. **Boundary value definition**: identify, if possible, boundary values for each input class

To illustrate the applicability of the above proposed procedure the following example will be given that sketches the procedure of test data elaboration for a benchmark test which is supposed to measure the *suitability* of a translation memory retrieval component.

STEP	TASK	DESCRIPTION
1	Problem domain definition	the system should provide translations to source language sentences that are identical or similar to those that were already translated; it should propose the translations in a way that the least necessary amendments have to be made
2	System functionality definition	the system performs segmentation, that is, separates different translation units the system stores translation units together with their source language equivalent the system retrieves identical or similar source language sentences and presents their translations the system recognises numbers and adapts them in the translation proposals
3	Error guessing	variation in sentence structure identical parts of sentences
4	Test cases definition	handling of variations in sentence structure split sentence with two segments into two separate sentences unite two separate sentences into one sentence change of sequence of main and sub-clauses handling of only identical parts of sentences deletion of sub-clauses
5	Input text examination	variation in formatting variation in brand names variation in type numbers variation in dates variation in acronyms
6	Input categorisation	handling of formatting handling of variable numbers handling of variable characters
7	Test cases definition	handling of formatting for formatted text strings: remove formatting, change formatting for non-formatted text strings add formatting handling of variable numbers change type numbers, date numbers handling of variable characters change names, acronyms
8	Boundary value definition	not identified in this case

Figure 84: Example for the Elaboration of Test Data

4.3.2 Types of Test Data in the Language Engineering Context

It has been pointed out before that test data can be distinguished in terms of their origin. In the language engineering context a frequent distinction of test data is between *test corpora*, *test suites* and *test collections*, which can be located at some point between the two poles of real text and artificially constructed test inputs. It is important to note that, though it is theoretically possible to draw clear lines between the three types of data, the boundaries become somewhat fluid in practical evaluation exercises. Apart from the origin of test items, major issues related to the elaboration of test data are representativeness, re-usability, cost, complexity of construction, and size.

In the following the three types of test data will be discussed considering the above issues, mentioning some of the advantages and drawbacks associated with each.

4.3.2.1 Test Corpora

Corpora consist of large quantities of naturally occurring machine readable text. The interest in corpora has grown with the power of increased computing capacity to process and store large amounts of data. Some of the best known corpora are the BNC (British National Corpus), the Brown Corpus of English, the Trésor de la Langue Francaise, and the bi-lingual (English-French) corpus drawn from the Canadian Hansard. In addition there are various initiatives that aim at the collection of what may be called 'general' corpora, such as the data collection initiative (DCI), launched by the Association for Computational Linguists, the Linguistic Data Consortium, and last but not least, the European Corpus Initiative.

The idea of making use of existing corpora for testing purposes is based on the assumption that, if the corpus is large enough, any linguistic or extra-linguistic problem that is of practical interest is bound to occur at least once. Yet, each text reflects properties related to the pragmatic background in which it was written. Therefore the problem of representativeness arises when making use of general test corpora. Specific evaluation scenarios mostly ask for a specific type of test input in terms of text type or language. Thus for evaluation, particular attention has to be paid to the question of what a particular corpus is representative for. The re-usability of test corpora is very high, since the same corpus may be used for testing various applications with a broad range of functionalities. The collection of corpora is a delicate matter in which ownership and copyright problems require careful consideration. The usage of existing corpora for testing purposes, however, ask for comparatively little money investment, since most of the corpora are available on FTP or WWW sites to a broad community. Another advantage of test corpora, is that the coverage of a test corpus is a matter of the size of the corpus rather than of a complex and costly construction of test inputs.

4.3.2.2 Test Suites

According to Balkan et al. (1995-2:3) test suites are artificially constructed sets of inputs that represent specific, pre-defined phenomena, which are systematically ordered to probe the system's behaviour with respect to these same phenomena. The elaboration of test suites involves the definition of the validity and nature of system inputs as well as the detailed consideration of its breath and depth of coverage. Balkan

et al. (1995-1:pp.42) present the following principles frequently adopted for test suite design:

* **one input sentence per phenomenon** allows the identification of the system's behaviour with respect to that phenomenon.
* **changing one parameter at a time** allows the identification of a system's problem with interacting phenomena.
* **less complex to complex phenomena** allows the identification of a system's limit of capacity.

According to Balkan et al. (1995-1:pp.44) advantages of test suites over test corpora mainly lie in the control over test data and its coverage: while the occurrence of critical phenomena in a test corpus is only accidental, a test suite allows the testing of a particular phenomenon in isolation, or combination, and allows a variation of parameters. Another advantage lies in the possibilities of the presentation of both test inputs and outputs by means of systematic annotation schemes. Moreover, the use of annotations, the classification of phenomena and the need for in-depth coverage of phenomena are some test suite characteristics that constitute a good basis for re-usability.

Yet problems related to the complexity of the construction of test suites for language engineering applications are broadly recognised: even at the level of syntactic phenomena, there are problems in identifying inputs which will test precisely what one wants to test, and once semantic, pragmatic or translation phenomena are taken into consideration, test suite design becomes a very delicate matter indeed. Closely related to the complexity of construction of test suites is their size and administration. King/Falkedal (1990) show that covering one phenomena per sentence, testing interacting phenomena and changing parameters, test suites can quickly become unmanageably large. Finally, given the complexity of construction and the effort of administering test suites, the costs for their design and usage are comparatively high.

4.3.2.3 Test Collections

In the classical sense, test collections consist of a set of inputs associated with a corresponding set of expected outputs and thus comes closest to the above SE definition of a test case. The major problem in elaborating test collections lies in the definition of expected sets of outputs, which mainly involve the definition of *correctness*. The MUC evaluations in Lehnert/Sundheim (1991) showed that in information retrieval, the area with the most prominent experience in developing test collections, the definition of the *correctness* of outputs for metrics such as 'recall' and 'precision' is straightforward. For translators' aids' systems or machine translation,

however, it is not always possible to define the *correctness* of outputs in exact terms. This is due to the fact that there is in most cases no one and only solution to a translation problem.

The most obvious advantages of test collections lie in the possibility to arrive at very detailed objective, numeric results of a system's capacity to deal with particular phenomena. The value of test collections and their re-usability to comparatively evaluate systems of the same type cannot be denied either.

The complexity of construction of test collections and the corresponding high costs, however, makes it hard to imagine their being constructed outside the evaluation guided research paradigm. Even for a minimal coverage of test cases, the size of test collections can become very large and hard to administer. In terms of re-usability, test collections are very specific to particular types of systems and thus need adjustment, when being applied to systems with a slightly different functionality. Nevertheless, are test collections in principal a very valuable source for further evaluation research.

The following table summarises the critical issues of the three types of test data in terms of their representativeness, re-usability, complexity of construction, cost and size.

ISSUES	TEST CORPORA	TEST SUITES	TEST COLLECTIONS
representativeness	critical for general corpora	depends on classification of phenomena	depends on classification of phenomena
re-usability	high	medium: mainly of annotations, classification of phenomena	critical: need major adjustment for different systems
complexity of construction	collection of corpora: medium (ownership/copyright) usage of corpora: low	high: one test input per phenomena; interacting phenomena; changing parameter	very high: same problem as test suites, **plus** definition of expected output
cost	low	high	very high
size	needs to be rather large	large	large

Figure 85: Overview of Critical Issues of Types of Test Data

To conclude the discussion of current practice of test data elaboration in the language engineering context, one may say that current approaches to language engineering test suite design could benefit from a more thorough consideration of the principles of equivalence partitioning and boundary value analysis to govern the size of the test data. Moreover it is envisaged that equivalence classes that are thoroughly defined for particular types of systems, for instance, spell checkers or translation memory systems,

could be to a large extent re-usable for different evaluation scenarios involving similar types of systems.

4.3.3 Parameters Determining the Selection or Elaboration of Test Data

So far different principles for test data elaboration have been discussed and the types of test data in the language engineering context presented. Yet which are the parameters that principally determine which type of test data should be used in a particular context? Balkan et al. (1994-1:pp.26) describe five parameters determining the conditions of testing that may have a great influence on the selection and/or elaboration of test data:

- black box vs. glass box situation
- availability and definition of pre-specified requirements
- definition and measurement criteria of an acceptance level
- types of texts involved
- languages and/or language pairs involved.

However, apart from the aspects presented by Balkan et al. there are further parameters that determine the selection/elaboration of test data. The following table attempts to present a more exhaustive list of parameters that influence the elaboration of test data in the language engineering context. Parameters are classified as belonging to the domain, the system, the evaluation or the administration category.

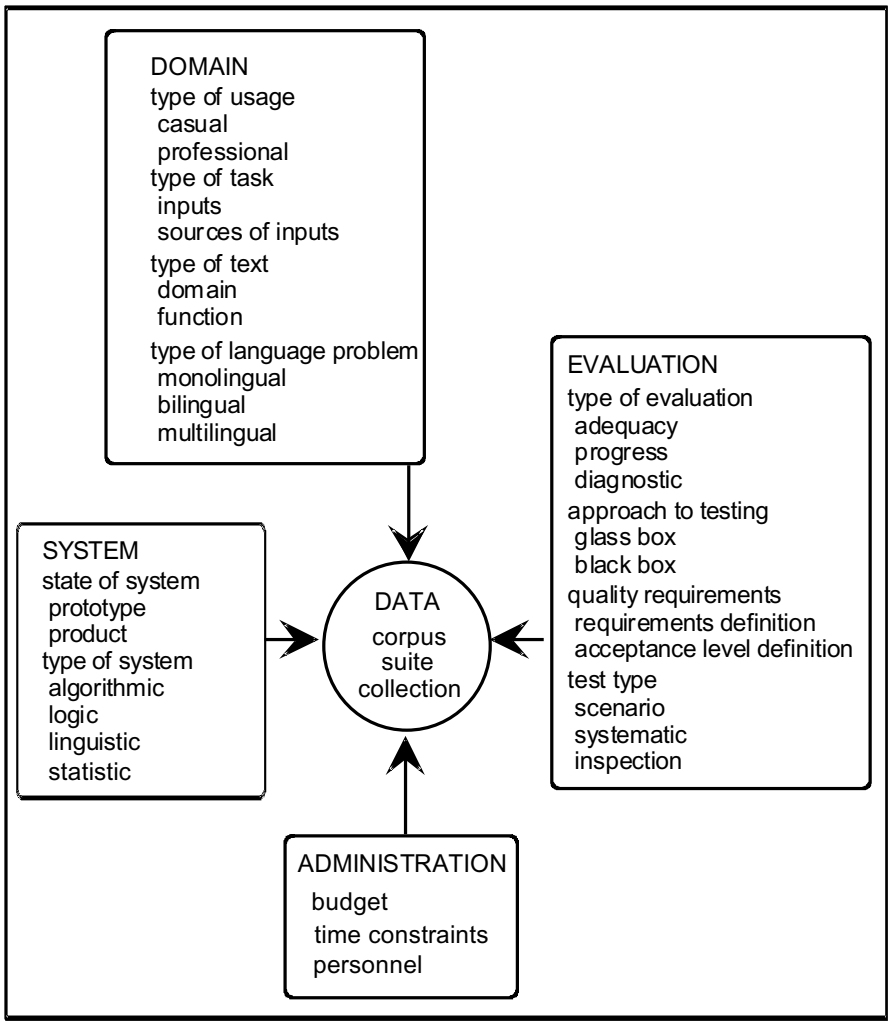


Figure 86: Parameters Determining Selection of Test Data

Considering the above parameters *before* entering a test makes sure that everything has been thought of properly and no unexpected conditions come up when it is too late to consider them. In the following, the parameters and their impact on the selection of test data will be discussed briefly.

4.3.3.1 Parameters of the Domain Category

On the domain side, the first parameter influencing the selection of test data is the type of usage: is the program used daily in a professional context or only on few occasions? While for casual usage, random data selection may be sufficient, for professional usage, the selection of test data must be driven by functional considerations.

Another factor related to the domain category is the type of task on which the test will be based. Only tasks should be selected for testing that represent a realistic application of either a casual or a professional type of usage. Tasks may include only one type of input or may be a complex combination of different types of inputs and outputs from different sources (human; programs/databases; devices). The elaboration of test data for complex tasks, in which inputs from various sources are needed, require careful planning including the preparation and/or customisation of resources.

Another parameter that is closely related to the type of task in the language engineering context is the type of text dealing with. Text types can be classified according to their domain and their function, where the domain of a text refers to the subject area it deals with, e.g. automobile, aircraft, business management etc., and the function of the text denotes its form and pragmatic function, e.g. letters, manuals, etc. Balkan et al. (1994-1:31) point out that for the elaboration of test data it is important to consider that "each text type has its own particular vocabulary and conventions, ranging from syntactic constructions through to formatting conventions."

As identified by Balkan et al. (1994-1:27) the type of language problem is a parameter that is of major importance for the elaboration of test data for language engineering applications. They distinguish between monolingual test suites in which the design of the test suite in the source language may be influenced by the expected results in the target language, and multilingual test suites in which the evaluator has to identify and exemplify correspondences across languages. According to Balkan et al. this may be achieved by identifying a core set of multilingual phenomena, which remain stable across languages, or by adapting the test suites to the specificity of each language, and state equivalencies where they happen to exist. Complex test tasks may require the language preparation and customisation of different resources that are consulted during the execution of the task.

4.3.3.2 Parameters of the System Category

For the elaboration of test data it is important to consider, the state of the system under testing. When dealing with a prototype of an early stage of the software life-cycle, test data has to be selected according to the restricted functionality of the prototype, whereas when dealing with an off-the-shelf product, the system can be tested against a realistic set of phenomena relevant to the specific product.

Another factor of the system category that is of great relevance for the selection of test data is the type of system, that is, whether dealing with systems based on algorithmic, logic, linguistic or statistic processes. While for a mere black box testing situation, the

selection of test data is mainly driven by parameters such as type of usage/tasks/text/language problem etc., a more informed test approach may also involve the consideration of the type of system when selecting test data. Functional knowledge of the type of system that goes beyond the recognition of mere user requirements, leads to the identification of phenomena that represent problematic cases for the type of processes involved. Test data that was developed in the awareness of the type of system and its functional properties is geared to that same type of system and it is questionable to which degree it is useful for systems of a different type. Balkan et al. (1994-1:35), for instance, discuss the usability of linguistic test suites for statistical based NLP systems and conclude that it is not clear whether linguistically based test suites are of use at all in the statistical context, since statistical systems do not work on linguistic rules. The development and validation of test data that is based on the recognition of the type of system certainly is a field in which more focussed research is compulsory.

4.3.3.3 Parameters of the Evaluation Category

A primary parameter of the evaluation category is the type of evaluation performed. For adequacy evaluation the selection of test data should most of all be driven by the requirements of the target group(s) under consideration, that is, parameters of the domain category have highest priority. In diagnostic evaluation, which aims at localising deficiencies, the selection of test data should be driven primarily by parameters of the system and evaluation category, that is, state of system, type of system, type of test, approach to testing etc.. The same is true for progress evaluation, in which successive stages of development of a system are compared.

As Balkan et al. (1994-1:26) point out, the approach to testing also greatly determines the selection of test data. Depending on whether evaluators have an extensive knowledge of the system internals or only have access to their user interfaces, they will adopt a different strategy of testing and will focus on different phenomena. Naturally glass box evaluation focuses on parameters of the system category, while in black box evaluation, the evaluator normally does not have access to information about the system structure but rather focuses on parameters of the domain category.

As identified by Balkan et al. the existence of a definition of quality requirements influences the elaboration of test data. From a pre-specified list of requirements, the evaluator can deduct which type of usage, tasks, texts and language problems are important to be covered by test data. The definition of measurement criteria of an acceptance level, furthermore, deal with the questions: what are the expected results of evaluation? what is the tolerance threshold in case of inadequate results? Answers to

these questions do not only help in the process of identification of relevant phenomena to be tested but also for the interpretation and assessment of results.

Finally the test type influences the choice of test data to a great extent: in a scenario testing environment, particularly in field tests, the realistic task asks for a selection of test data that is both typical for the specific environment and representative for the type of usage. In most cases, test data for scenario tests in the language engineering area, therefore, are either extracted from existing corpora or at least based on corpora, integrating additional typical or problematic phenomena. For systematic testing, test suites and collections can be elaborated that integrate both typical and problematic phenomena for the process under testing.

4.3.3.4 Parameters of the Administration Category

The discussion of the three types of test data already pointed to there being a great difference in cost between test corpora, test suites and test collections. Naturally, the budget of evaluation to some extent governs the selection of data: a low budget may argue for test corpora or small test suites, a larger budget allows the development of test collections. In addition the time constraints of evaluation are often responsible for choosing one particular type of test data. Corpora are more quickly provided than test suites or even test collections are elaborated. Finally the personnel involved in the evaluation procedure determines to some extent what type of test data will be used. While the selection of corpora does not ask for particular expertise, the elaboration of test suites, and even more so of test collections asks for a certain expertise in the field.

To conclude, the above section on test data elaboration could but outline the general principles and considerations that are relevant to the elaboration of test data. The discussion showed that the complex and costly elaboration of test data pointed towards the tension between the construction of general, re-usable test data and the precept that each evaluation is specific in the sense that it is carried out for a particular reason, for a particular system, in a particular environment - a problem that is also well-known with respect to evaluation methodologies. Whether or not, and to which extent existing data can be used for a particular environment always has to be decided for each context separately. A very important issue, therefore, is to make the data available to all potentially interested people on FTP or WWW sites.

4.4 Experiences and Results of Testing in Evaluation

Each evaluation scenario has its own background which will determine the testing procedure to a great extent.

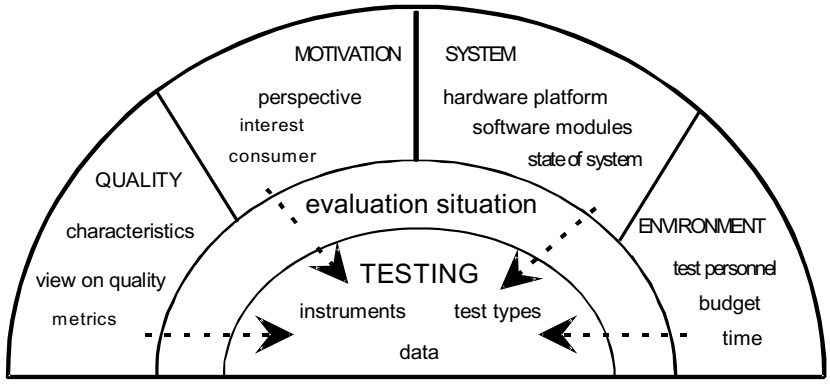


Figure 87: Factors Influencing Evaluation

Among the most general factors that influence the way tests are performed are the motivation behind evaluation, the system and its parameters, the evaluation environment, and, finally, the quality requirements that need to be tested.
Galliers/Sparck-Jones (1993:186) consider motivation to cover three factors, that is, perspective, interest and consumer of the evaluation:

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

(ii) the *interest* taken in the evaluation denotes the view taken on the evaluation process, i.e. even for the same type of evaluation, a developer may have a totally different view on what needs to be considered than the funder of a project, who will again put different foci than user-organisations etc.

(iii) the *consumer* of the evaluation report denotes whether managerial, scientific, practical or implementation related aspects are focused during evaluation and reporting.

The settings of system parameters such as hardware platform, software modules and the state of the system, that is, whether the system is a prototype, ß-version or product, are givens and determine both procedure and results of the testing process. For instance, if a system prototype is evaluated, metrics measuring *efficiency* often cannot be applied, since the system is not fully implemented, while for product evaluation, *efficiency* is one of the most important quality characteristics.

The evaluation environment is determined by the test personnel, the size of the budget and the amount of time invested. Falkedal (1991:20) divides test personnel into two major groups: (i) experts that function as evaluators during evaluation and (ii) users that function as subjects of tests. User-oriented testing should be performed by someone who was not involved in the development of the software. A certain knowledge of software engineering principles, and of both the practical and the computational side of the application under testing is, however, very useful, if not indispensable. Depending on the quality characteristics and the metrics applied, as well as on the test type and instruments, different types of users - students, professional users - can or rather need to be used as subjects. According to Falkedal (1991:pp.21) crucial qualifications required by test persons are (i) objectivity, (ii) representativity, and for all language engineering applications (iii) language proficiency. At the same time she points out that striving for objectivity runs the risk of making evaluations and tests using purportedly unbiased evaluators and test persons into purely artificial events whose results are likely to be insufficiently informative about or representative of how a system would be judged by its end-users, once they had become accustomed to its peculiarities.

The evaluation budget is naturally the most decisive factor, when it comes to selecting evaluators, subjects, test types and instruments as well as when determining the time that can be invested into evaluation. It is important to note here that, in case of a limited evaluation budget, it is advisable to reduce the number of metrics that will be tested and to select less expensive instruments, rather than to reduce the number and qualification of test personnel. While a limited number of metrics only reduces the scope of evaluation, savings in the area of test personnel may question the reliability and validity of test results.

The selection and determination of quality characteristics and attributes for a particular evaluation process depend on all of the above mentioned factors, that is, motivation, system parameters and environment. While, for instance, from a financial perspective, *efficiency* is the most important quality characteristic, an administrative perspective will rather focus on *inter-operability* and *usability*. The quality of a translators' aids' system can be determined at different levels of detail, depending on the consumer of the evaluation report and the interest behind evaluation. Black box evaluation, normally performed in evaluation preceding purchase decision, asks for less differentiated quality considerations, while glass box evaluation, normally performed in evaluation supporting development, asks for a differentiated quality report, determining clearly whether test results are related to the system interface, its system functionality, or the data offered, for instance by a termbank. The situation in which

evaluation is performed determines the evaluation procedure, that is, whether evaluation is performed preceding purchase decisions on behalf of translation industry, or whether evaluation is performed during translation system development to support the development process.

Considering the two evaluation situations, that is, evaluation preceding a purchase decision and evaluation supporting software development, the above factors play an important role. The following table lists the major factors that influence the evaluation procedure and their effect on testing:

FACTORS	EVALUATION PRECEDING PURCHASE DECISIONS	EVALUATION SUPPORTING DEVELOPMENT
perspective	high priority to financial perspective;	task-oriented perspective, that is, how can system perform given tasks
interest	to find out which system under evaluation suits the given environment best	to improve the software in order to meet user requirements
consumer	mostly management and/or users	system engineers; scientists developing solutions to engineering problems; funding organisations
state of system	off-the-shelf products	from prototypes or individual modules of prototypes to ß versions of systems
evaluation environment	low budget little time	testing at different stages of software life-cycle
quality	focus on value and evaluation relevant metrics;	focuses on qualitative rather than quantitative measurement;

Figure 88: Factors that Influence the Testing Process

From the above table it follows that testing supporting development should be broader in scope and more exploratory in nature than in evaluation preceding purchase decisions. It should not only assess pre-defined metrics but develop additional ways to assess the quality of a system during the testing procedure; note observations about the nature of the system; make proposals for modification, and try to find out where problems lie. A good example for testing supporting the development process is the testing of the Translator's Workbench in the ESPRIT projects TWB I (2315) and TWB II (6005), which was conducted by different user organisations under the supervision of the author of this thesis. For experiences and results of testing supporting the development process the reader is referred to Höge/Hohmann/Le-Hong (1995:pp.168-173). An excerpt of the TWB result report can be found in appendix 1. In the context of this thesis, testing in evaluation focuses on evaluation preceding purchase decisions, since it is there, where the new approaches can be applied.

4.4.1 The Testing Context

Considering the factors that were discussed above, the following picture can be drawn of the testing process which will be described below:

- In the context of this thesis, the perspective was task-oriented and the interest behind testing was to prove that the framework developed for user-oriented evaluation actually works. The evaluation report in this thesis addresses first and foremost a scientific audience, and secondly user-organisations who should be able to use the evaluation framework.
- The systems tested were Trados TWB for Windows, ß version (system x) and IBM TM/2 version 1.0 (system y).
- As evaluator, the author of this thesis is knowledgeable in the areas of software development and requirements engineering and can be considered expert with experience in testing and as user-representative in the TWB projects. The subjects used for testing were advanced students of translation who learned to work with the systems in the context of a computer aided translation course. The testing budget was low and the time involved in evaluation preparation and testing along the pre-defined framework developed in this thesis were around 50 days.
- There was a focus on quality characteristics related to usability and efficiency, but also covering reliability issues. Only value relevant metrics were applied, evaluation relevant metrics identified after testing.

Evaluation preparation was performed along the lines presented in chapter 3. The task description in figure 28 formed the basis for the development of metrics for testing. The tasks under testing were:

- translation memory preparation (t_1)
- translation memory and termbank retrieval (t_2)
- updating translation memory databases(t_3)
- updating termbanks(t_4)

In short, for each of the above tasks, the qualitative aspects presented in figures 61-66 were applied as described in chapter 3 leading to a list of 87 metrics and corresponding scales which can be found in appendix 2. Value functions were developed for each of the metrics, defining how the value of an attribute increases/decreases with a specific scale value, leading to **v(x)** and **v(y)** for each of the metrics. For each metric belonging to (t_1) - (t_4), it was decided which test type would be most appropriate as described in section 3.2.1.4 (test model). Along the process defined in this section, for each task under evaluation, the relevant metrics first were applied in *feature inspection*. For those metrics for which values could not be obtained through *feature inspection*, other

test types were applied. Those metrics for which **v(x)** and **v(y)** were **not** identical after the tests, were identified as evaluation relevant (rel).

To illustrate this, the following excerpt of appendix 2 will be provided. For each quality subcharacteristic to be tested within each task a separate table is presented which covers the above described test data:

Installability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
installation program	binary	1	100	1	100	task	68	-
time needed for installation	ratio v(60)=0	30 Min	50	50 Min	16,7	task	69	✓
installation without knowledge of operating system possible?	binary	1	100	1	100	task	70	-

Figure 89: Excerpt of Result Report (t_2) from Appendix 2

As appendix 2 shows, the major test type performed in (t_1) was feature inspection with 13 of 18 metrics (No. 1-10, 14-16). The programs tested in (t_1) fulfilled the basic requirement for benchmark testing, that is, being batch programs they work independent of human interaction. Consequently three metrics (No. 11-13) could be applied in benchmark testing. Two metrics (No. 17-18) were applied in task-oriented testing.

Task 2 (t_2), that is, translation memory and termbank retrieval, was exhaustively tested (52 metrics) with all test types. The major test type applied for (t_2) was task-oriented testing (28 metrics), followed by feature inspection (19), interface-oriented testing (10), scenario testing (7), and, last but not least, benchmark testing (2). To increase reliability and validity, many metrics were applied in more than one test:

- 4 were tested in both task- and interface-oriented testing.
- 6 of 7 metrics applied in scenario testing were also applied in task-oriented testing, 1 as well in feature inspection.
- 3 metrics were applied in scenario testing as well as in task- and interface-oriented testing.

Task 3 (t_3), that is, updating translation memory databases, covered 9 metrics, 8 of which were tested by means of task-oriented testing, 4 by interface-oriented testing, 1 by feature inspection, 1 by scenario testing and none by benchmark testing. Again, some of the metrics were applied by a combination of task-oriented testing and either interface-oriented or scenario testing.

Task 4 (t_4), that is, updating termbanks, covered 6 metrics, 4 of which were tested in task-oriented testing, and 2 in interface-oriented testing. 1 metric was tested in both test types. None was tested by means of feature inspection, benchmark or scenario tests.

It is very important to note that the primary focus of the tests performed in the context of this thesis was to *assess the applicability of the testing approach for evaluation preceding purchase decisions* rather than to investigate the quality of the systems under evaluation. For a functional description of different translators' aids' systems see Spies (1995). The following discussion of testing experiences will concentrate on issues related to the preparation and performance of the tests. For those interested in how the two systems performed, the results presented in appendix 2 will provide the details. Among all test types, it is the scenario test that will be presented in most detail, to point to the problems and difficulties which this kind of testing might incur.

4.4.2 Experiences with Feature Inspection

In context of the tests performed for this thesis, *feature inspection* played a major role. No less than 35 of 87 metrics (39.9 %) could be applied solely by going through the system documentation in feature inspection. In most cases the values obtained could be measured on a binary scale (23), in several cases (12) also on binary nominal scales. Whenever the nominal attributes of a binary nominal scale had equal weight, the majority voting rule was applied, that is:

(100 ÷ number of possible attributes) number of actual attributes

Whenever the nominal options of a binary nominal scale were not equal in weight, weights had to be determined for the options, by dividing 100 among the options according to their importance, and summing the score of all options. The following figure provides an example for both calculations.

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type
recognition of special text elements:	binary nominal		7/7 = 100		7/7 = 100	insp
proper names		1		1		
codes		1		1		
numbers		1		1		
dates		1		1		
currencies		1		1		
tables		1		1		
figures		1		1		
selection of source text segment:	binary nominal		100		80	insp
automatic	80	1 x 80		1 x 80		
manual	20	1 x 20		0		

Figure 90: Examples for Calculation of v in Binary Nominal Scales

As the results in appendix 2 show, only 19 of 35 metrics applied during feature inspection proved to be *evaluation relevant*. In other words, in 16 cases both alternatives under evaluation scored the same on a given feature or metric. While in evaluation supporting development, each metric is relevant to the evaluation procedure, in evaluation preceding purchase decisions only these 19 evaluation relevant metrics will be part of the final assessment calculation which will be presented in chapter 5.

As pointed out before *feature inspection* can only lead to results with respect to the question whether or not specific features are present and not **how well** the features are implemented. This is subject to systematic and scenario testing.

4.4.3 Experiences with Systematic Testing

In the testing approach advocated in this thesis, systematic testing covers task-oriented, interface-driven and benchmark testing. Among the three sub-test types, most metrics were applied during task-oriented testing (42), followed by interface-driven testing (16) and, last but not least, benchmark testing (5). To validate critical results, some of the metrics (10) were applied in both task- and interface oriented testing.

Before performing systematic testing, the testbed had to be prepared in terms of test data. Specifically (t_1) and (t_2) asked for detailed preparation of test data to guarantee satisfactory results. Tasks (t_3) and (t_4) are concerned with updating of existing databases. There is no big difference as to which type of data to be updated. Consequently the data for (t_3) and (t_4) can be quite easily developed on an ad hoc basis during the tests. The preparation of test data followed the principles discussed in section 4.3. In the following figure the parameters relevant to data elaboration are described and their effect on the selection or elaboration of test data is indicated.

PARAMETER	DESCRIPTION	EFFECT ON ELABORATION OF TEST DATA
DOMAIN CATEGORY		
type of usage	professional translation context	functional test data selection
type of task	(t_1) translation memory preparation (t_2) translation memory and termbank retrieval	-> test data extracted from real text corpus ->test suites
type of text	(t_1) real text corpus for alignment of different domains and functions (t_2) real new version of document; test suite for more exhaustive coverage	(t_1) Setup 1: Domain: automobile industry Function: manual Setup 2: Domain: politics function: letter (t_2) Setup 1: Domain: automobile industry Function: manual Setup 2: Test suite
type of language problem	(t_1) alignment with different language directions (t_2) retrieval of similar texts	(t_1) Setup 1 Alignment de-en Setup 2 Alignment en-de (t_2) Setup 1 and 2 Retrieval in de-en translation
SYSTEM CATEGORY		
state of system	product	typical test text test suites
EVALUATION CATEGORY		
type of evaluation	evaluation preceding purchase decision	Test corpora Test suites
approach to testing	black box	Usage-oriented selection of test data
availability of quality requirements definition	Metrics developed in evaluation preparation phase	• definition of 54 metrics; • acceptance level definition available
test types	1. task-oriented testing 2. interface-driven testing 3. benchmark testing	1+ 2: Retrieving aligned text in TM database, using test text and terminology; 3. alignment benchmark retrieval benchmark
ADMINISTRATION CATEGORY		
budget	medium	text corpus, small test suite
time constraints	one week for elaboration of test data	Analysis of text corpus and selection of test text with representative characteristics, develop test suite that covers most important phenomena.
personnel	translator/computer linguist	Black box view, since system internals are not known.

Figure 91: Parameters and their Effect on Test Data Elaboration

As the above table shows, test data had to be prepared for tasks (t_1) and (t_2).

- For (t_1), that is, TM preparation two setups were used. Setup 1 included the alignment of a German car manual with its English translation. This alignment could later also be used as the basis TM for testing (t_2), that is, TM retrieval. Setup 2 covered an English letter from the EU translation department and its German translation. The two different setups for (t_1) were used in order to find out, whether both systems could handle both situations equally well.
- For (t_2) two setups were used. Setup 1 covered a real text: The new version of the car manual that was aligned in (t_1) was translated, making use of the TM aligned in (t_1). For this purpose the results of the alignment in (t_1) were imported into the translation memory database as the basis for retrieval of the new version. Setup 2 covered a test suite which was developed in order to assess a broader scope of retrieval possibilities (for more details see Experiences with Benchmark Testing).

The car manual texts for (t_1) and (t_2) were from a text corpus which was provided for testing by Mercedes-Benz, consisting of 6 repair manuals in an old and new version in German and English. In order to check the appropriateness of the manuals for (t_2), that is, translation memory retrieval, the German old-new pairs were compared by means of the WinWord document version comparison function. appendix 3 provides the results of the text analyses in tabular form. Text pairs 1, 5 and 6 showed a high occurrence of the above characteristics and would, therefore, have been appropriate as test data. The parallel text that was used for (t_1) and (t_2) was text pair 1 , that is, the old version of AR 27 in German and English for (t_1), and the new German version of AR 27 for (t_2). The following table is an excerpt of appendix 3 and presents the results of the text analysis. It shows the phenomena covered by the text, with the differences between the two versions written in capital letters.

TEXT ANALYSIS TEXT PAIR 1 MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION AR27	OLD VERSION AR27
numbers in identical segments	1. GETRIEBE 722.6 2. GETRIEBE 722.620/621/622	1. GETRIEBE 722.3/4/5
identical parts of sentences	1. Ölstand nochmals PRÜFEN 2. Eine zu kleine BZW. zu große Ölmenge beeinträchtigt die Funktion des Getriebes 3. Bei kaltem Getriebe muß DIE ÖLSTANDSANZEIGE zwischen der "min." und "max." -Markierung, 25° (GETRIEBEÖLTEMPERATUR) liegen. 4. Bei betriebswarmen Getriebe muß DIE ÖLSTANDSANZEIGE an der "max"-Markierung, [80°] (GETRIEBEÖLTEMPERATUR) anliegen 5. Getriebeöl (AFT) nach Betriebsstoff-forschriften-Blatt NR. 236.10	1. Ölstand nochmals KONTROLLIEREN 2. Eine zu kleine, SO WIE EINE zu große Ölmenge beeinträchtigt die Funktion des Getriebes 3. Bei kaltem Getriebe (GETRIEBEÖLTEMPERATUR CA 30°) muß BEI RICHTIGEM ÖLSTAND DIE ANZEIGE zwischen der "min." und "max" -Markierung LIEGEN (BILD 3). 4. Bei betriebswarmen Getriebe (GETRIEBEÖLTERMPERATUR CA 80°) muß BEI RICHTIGEM ÖLSTAND DIE ANZEIGE an der "max"-Markierung anliegen (BILD 3). 5. AFT-ÖL nach Betriebsstoff-Vorschriften Blatt 236.4/6/7
left out/added segments	1. Das Fahrzeug muß waagrecht stehen 2. Getriebe auf Dichtheit prüfen 3. Bei Ölverlust Ursache ermittlen 4. Getriebeöl einfüllen 5. Ölmeßstab [(6)] bis zum Anschlag einstechen UND wieder herausziehen, Ölstand ablesen 6. Betriebsstoff-Vorschriften	1. Das Fahrzeug muß ZUR ÖLSTANDSKONTROLLE waagrecht stehen 2. Getriebe VOR ÖLSTANDSKONTROLLE auf Dichtheit prüfen 3. Bei größerem Ölverlust Ursache ermitteln 4. Getriebeöl BEI LAUFENDEM MOTOR einfüllen 5. Ölmeßstab (6) bis zum Anschlag einstechen [,] wieder herausziehen, Ölstand ablesen 6. AFT-ÖL NACH Betriebsstoff-Vorschriften BLATT [236.4/6/7]
identical individual terms	1. Getriebe 2. Handpumpe 3. Trichter	
identical segments	1. Sicherheitsvorschriften bei laufendem Motor beachten 2. Ggf. berichtigen 3. Zuviel eingefülltes Getriebeöl unbedingt ablassen oder absaugen,	

Figure 92: Text Analysis of Test Data used for (t_1) and (t_2)

Experiences with Task-oriented Testing

In task-oriented testing, the evaluator performed the tasks (t_1) to (t_4) with both systems and, where relevant, applied the metrics presented in appendix 2. The results of the tests were then noted, and the values for each system v(x) and v(y) calculated.

Task 1, that is, translation memory preparation was realised in form of batch programs by both systems under evaluation (system x: Trados TAlign; system y: IBM ITM). Due to only minor interaction while performing the tasks, the only metrics applied in task-oriented testing for this task were those related to *installability* (No. 17,18), measuring whether installation programs were provided and installation was possible without knowledge of the operating system.

Task 2, 3 and 4, that is, translation memory and termbank retrieval, and updating translation memory and termbank are highly interactive and therefore particularly promising for *task-oriented testing* of the two systems. Task 2 covered the translation of the new AR27 text with the TM produced of the alignment of the old AR27 text that was the basis of (t_2). During translation, the different metrics were applied, the results noted in the result reports presented in appendix 2 and the values v(x) and v(y) calculated for each metric. Again, only metrics with non-identical values were noted as evaluation relevant. For (t_3) it was investigated whether and how TM databases could be modified by users. The tests showed that only system x allowed the modification of TM databases after translation. Consequently v(y) was 0 in all related metrics. For (t_4) the modification procedures of the terminology database were investigated. The metrics which were considered relevant and listed in appendix 2 were applied during this task and their outcome noted on the result reports. The ad hoc generation of test data according to the "WHAT HAPPENS IF" principle proved to be non-problematic.

To sum up, most metrics applied during task-oriented testing were measured on binary scales (26). Only few were measured on ratio (5) and binary nominal (3) scales. Ordinal scale metrics (8) that were related to user likes and dislikes were also subject to later scenario testing in order to check their reliability. 31 of 42 metrics applied in *task-oriented testing* proved to be *evaluation relevant* (73,8%). This shows, that though frequently the documented features of two systems are similar (see results of *feature inspection*), the way they are implemented still show differences that may have a decisive effect on the evaluation outcome.

Experiences with Interface-driven Testing

Interface-driven testing was performed to (i) examine critical issues of the software, and (ii) to make sure that the coverage of the tests is sufficient. For this purpose the menu options presented by the two systems were followed one after the other and the metrics of appendix 2 applied where necessary. The quality characteristics primarily investigated were *security*, *fault tolerance* and *stability*. Furthermore the *usability* of operation control mechanisms was investigated by *interface-driven testing* in combination with *task-oriented* and *scenario testing*, to provide the possibility to count

steps or consider the similarity of operation control mechanisms. The fact that 13 of 16 metrics applied during *interface-driven testing* proved to be *evaluation relevant* shows that interface-driven testing is a means to uncover differences in systems which may not be found by means of other test types but should be considered in evaluation.

Experiences with Benchmark Testing

Among the 87 metrics applied during the tests, only 5 were measured by *benchmark tests*. This low figure points to the fact, that, in the context of translators' aids' systems, it is difficult to identify functions of the system that are independent of user input. In the language engineering context, *benchmark tests* measure mainly response time or performance of systems in terms of recall and precision.

In **Task 1** (t_1), being concerned with a batch program for the alignment of parallel translations, a *benchmark test* could be performed, applying the metrics

- **alignment rate**: number of aligned segments / number of segments;
- **alignment success rate**: number of correctly aligned segments / number of aligned segments:
- **total success rate**: number of correctly aligned segments / number of segments.

Comparing alignment *benchmark test* results is a delicate matter indeed. Even if the testbed is exactly the same for the different systems under testing, it may favour one system over the other, since the characteristics of the test data are, probably by chance, those that are considered by the system. This is due to the fact that in black box evaluation, details about the nature of the programs are not known. Choosing different test data, the results may be totally different. Therefore, it is of utmost important to choose exactly the type of test data that is relevant to the particular customer of the evaluation procedure and to refer test results directly and only to the specific test bed. Making use of two different setups as described above, the *validity* of test results is checked. The two text that were used for the benchmark test in setup 1 and 2 will be presented in appendix 3. The following table lists the characteristics of the texts in the two setups together with the benchmark test results:

	SETUP 1	**SETUP 2**
provider of text	Mercedes-Benz	EU Translation Service SdT
function	car manual	letters
domain	automotive engineering	politics
number of segments	SL: 39; TL: 39	SL: 33; TL: 37
characteristics of text	1. simple sentences; 2. most sentences separated by paragraphs; 3. many numbers and acronyms; 4. many singular terms 5. abbreviations (ca.) and dates.	1. difficult formatting due to footnotes, pre-defined initials and fixed paragraphs; 2. missing parts in the English version (no date); 3. abbreviations (Nr.) or dates (20. Juli); 4. 1:2 and 2:1 translations.
experience	The two systems could cope with the text easily. **System 1** aligned 36 segments, 33 of which correctly, leading to **alignment rate value**: 92.30 **alignment success value**: 91.66 **total success value**: 84.61. **System 2** aligned 37 segments, 34 of which correctly, leading to **alignment rate value**: 94.87 **alignment success value**: 91.89 **total success value**: 87.17.	The two systems aligned almost the same amount of segments, while the correctness of alignments greatly differed between the systems. **System 1** could align 27 segments, 26 of which correctly, leading to **alignment rate value:** 81.81 **alignment success value:** 96.29 **total success value:** 78.78. **System 2** aligned 24 segments, 15 of which correctly. This leads to **alignment rate value:** 72.72 **alignment success value:** 62.49 **total success value:** 45.45. The system was led astray almost from the very beginning, producing incorrect alignments up to almost half of the text, from where on it aligned correctly.

Figure 93: Experiences with Benchmark Tests for (t_1)- TM Preparation

The results of the above two setups support the view that the simpler the text type the better the alignment result and as such validates the benchmark procedure. Car manuals is a text type that can easily be handled by both alignment systems alike, while complex text types such as letters need far more post-editing. The results show that while system 1 is at least partly capable of solving the problems posed in letters, it does not make any sense to apply system 2 for more complex text types. The calculation of v(x) and v(y) involved the weighting of the importance of handling the two types of text in the domain by dividing 1 up among the two setups. In other words, the results of setup 1 were multiplied by .80 and the results of setup 2 by .20 because in the context of an industrial translation department such as the Mercedes-Benz translation department, the translation of manuals is far more important than that of letters.

In **Task 2** (t_2), that is, translation memory and termbank retrieval, two types of benchmark tests were applied, that is, (i) measuring the quality of the retrieval

component of the translation memory, most interestingly, the fuzzy match component, and (ii) measuring time behaviour for translation memory retrieval.

For the first type of benchmark, that is, measuring the quality of the retrieval component, again two setups were identified to increase the *validity* of test results. As described before, for **setup 1** the test data was taken from a real *text corpus*, that is, the new version of the AR 27 car manual provided by Mercedes-Benz, which will be included in appendix 3. For **setup 2** a *test suite* was generated that was supposed to probe the translation memory retrieval system for specific phenomena.

In **setup 1**, which is concerned with a *real* text from a *real* environment, the definition of *value relevant* and *valid* metrics for the translation memory retrieval component proved to be difficult. In other words, which aspects involved in the retrieval process allow sensible measurement? It has to be kept in mind that the principal goal of making use of fuzzy matches is that it has to be *less effort* to evaluate the *suitability* of the proposals for the given context than to translate the segments manually. Consequently, the first idea for a metric was to assess the effort by counting the keystrokes necessary to make use of a fuzzy match proposal. It would also be possible to define acceptance levels and develop value functions for this metric. However, on closer examination, it proved not to be generally *valid*, since (i) it depends on a system's user interface, that is, on its means of operation control; and (ii) it is sensitive to one's editing habits, that is, whether using shortcuts etc. Consequently, counting keystrokes would be a combined measure of the *suitability* of the retrieval component and the *operability* of the system interface and not totally independent of user input.

The second possibility identified was to measure *recall quantity*, that is, how frequently does the system come up with a fuzzy match proposal for a SL segment. The quantity of fuzzy match proposals, however, depends on the minimum match value which is either internally set by the system (IBM TM/2) or can be set by the user (Trados TWB4W). In other words, if the minimum match value in the Trados system is set to 30%, more matches with presumably a lower quality will be retrieved than if it is set to 60%. This fact makes the comparison of results of recall quantity not *reliable*.

Another idea was to count the *ratio between words that can be taken over directly and the total number of words per segment*. Again, it was found that the number of words is no clear indicator as to how useful a fuzzy match proposal is, because functions words, for instance, pose no problems for the translator while terms do. The *validity* of this measure would, again, be rather low. As pointed out before, the quality of a TM retrieval component depends on its capability to recognise identical or similar text.

Consequently, the only sensible way to measure the quality of the translation memory retrieval component in a real setup, finally, proved to be to describe the phenomena that are present in the test texts and measure the *quantity of recall of text segments* as they were relevant when comparing the old and the new versions of the car manual. The following table represents the phenomena that occurred in the test texts, the relevant text segments and the results of both systems under testing.

RESULTS RETRIEVAL BENCHMARK RECALL OF REAL PHENOMENA TASK 2 SETUP 1			
TYPE OF PHENOMENA	TEXT SEGMENTS	RECALL SYSTEM 1	RECALL SYSTEM 2
numbers in identical segments	1. GETRIEBE 722.6 2. GETRIEBE 722.620/621/622	1 1	0 0
identical parts of sentences	3. Ölstand nochmals PRÜFEN 4. Eine zu kleine BZW. zu große Ölmenge beeinträchtigt die Funktion des Getriebes 5. Bei kaltem Getriebe muß DIE ÖLSTANDSANZEIGE zwischen der "min." und "max." -Markierung, 25° (GETRIEBEÖLTEMPERATUR) liegen. 6. Bei betriebswarmen Getriebe muß DIE ÖLSTANDSANZEIGE an der "max"-Markierung, [80°] (GETRIEBEÖLTEMPERATUR) anliegen 7. Getriebeöl (AFT) nach Betriebsstoff-forschriften-Blatt NR. 236.10	1 0 0 1 1	1 0 0 0 0
left out/added segments	8. Das Fahrzeug muß waagrecht stehen 9. Getriebe auf Dichtheit prüfen 10. Bei Ölverlust Ursache ermittlen und beseitigen 11. Getriebeöl einfüllen 12. Ölmeßstab [(6)] bis zum Anschlag einstechen UND wieder herausziehen, Ölstand ablesen 13. Betriebsstoff-Vorschriften	1 1 1 1 1 0	0 0 1 0 0 0
identical individual terms	14. Getriebe 15. Handpumpe 16. Trichter	1 1 1	0 1 0
identical segments	17. Sicherheitsvorschriften bei laufendem Motor beachten 18. Ggf. berichtigen 19. Zuviel eingefülltes Getriebeöl unbedingt ablassen oder absaugen,	1 1 1	1 1 1
TOTAL RECALL OF PHENOMENA		16/19	6/19
FUNCTION VALUE FOR RETRIEVAL BENCHMARK SETUP 1		84.21	31.57

Figure 94: Benchmark Test Results (t_2) Setup 1

In short, for system 1 the default minimal match value was set to 30%. With this setting it could retrieve all in all 27 matches. Among the 27 matches 16 corresponded to the text segments identified in the text analysis. The remaining 11 matches showed similarity on the term level. For system 2 no minimum match value could be set by the user. It could retrieve 6 of the 19 text segments, leading to the followinig values:

v(x) = 84.21 v(y) = 31.57

For **setup 2** a test suite was developed in accordance with the procedure of test data elaboration presented in section 4.3. The following table shows the steps that were involved in the definition of the test suite based on the analysis of texts from Mercedes-Benz, which can be found in appendix 3.

STEP	TASK	DESCRIPTION
1	Problem domain definition	• the system should provide translations to source language sentences that are identical or similar to those that were already translated; • it should propose the translations in a way that the least necessary amendments have to be made
2	System functionality definition	• the system performs segmentation, that is, separates different translation units • the system stores translation units together with their source language equivalent • the system retrieves identical or similar source language sentences and presents their translations
3	Error guessing	• variation in sentence structure • only identical *parts* of sentences
4	Error guessing: Test cases definition	handling of only identical *parts* of sentences: • deletion of sub-clauses (1) handling of variations in sentence structure: • split sentence with two segments into two separate sentences (2) • unite two separate sentences into one sentence (3) • change of sequence of main and sub-clauses (4)
5	Input text examination	• variation in formatting • variation in brand names • variation in type numbers • variation in dates • variation in acronyms
6	Input categorisation	• handling of variable characters • handling of variable numbers • handling of formatting
7	Test cases definition	handling of variable characters • change names (5) • change acronyms (6) handling of variable numbers: • change numbers (7) • date numbers (8) handling of formatting: • for formatted text strings: remove formatting (9) • change formatting (10) • for non-formatted text strings add formatting (11)

Figure 95: Test Suite Generation for Retrieval Benchmark Setup 2

While in setup 1 the retrieval component could only be tested with respect to the 19 *text segments* present in the test text, in **setup 2** the system could be tested with respect to the *quantity of recall* for all of the 11 *test cases* that were defined above. For this purpose a sentence that was stored in the TM database was altered according to the pattern described in the test cases and the success for retrieving the original stored

sentence was measured in boolean terms, that is, whether or not the original sentence was retrieved. To improve the *reliability* of the test, the same test case was, if possible, applied with 5 segments, leading to a test suite with 46 test items. For the test it was counted, how frequently the system successfully retrieved the original segment. The following figure presents the results of the benchmark test.

RESULTS RETRIEVAL BENCHMARK RECALL OF TEST CASES TASK 2 SETUP 2			
TEST CASE	TEST ITEM	RECALL SYSTEM 1	RECALL SYSTEM 2
deletion of sub-clauses	1. Ölstand im automatischen Getriebe prüfen.	1	0
	2. Bei größerem Ölverlust Ursache ermitteln.	1	0
	3. Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken.	1	0
	4. Ölmeßstab (6) bis zum Anschlag einstecken.	1	1
	5. Sicherungsstift (6b) einsetzen, bis er einrastet.	1	0
split sentence with two segments into two separate sentences	6. Ölstand im automatischen Getriebe prüfen. Ölstand im automatischen Getriebe richtigstellen.	1	1
	7. Bei größerem Ölverlust Ursache ermitteln. Bei größerem Ölverlust Ursache beseitigen.	1	1
	8. Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken. Beide Teile entfernen und Verschlußhebel (6a) öffnen.	1	1
	9. Ölmeßstab (6) bis zum Anschlag einstecken. Ölmeßstab (6) herausziehen, Ölstand ablesen.	1	1
	10. Verschlußhebel (6a) schließen. Sicherungsstift (6b) einsetzen, bis er einrastet.	1	0
unite two separate sentences into one sentence	11. Fahrzeug zur Ölstandskontrolle waagrecht stellen (1) und Getriebe vor der Ölstandskontrolle auf Dichtheit prüfen.	1	1
	12. Motor laufenlassen (3.1) und Verschlußhebel (6a) öffnen.	1	0
	13. Ölmeßstab (6) herausziehen und mit fusselfreiem Tuch abwischen.	1	0
	14. Bei kaltem Getriebe (Getriebeöltemperatur ca. 30°C) muß bei richtigem Ölstand die Anzeige zwischen der "min." und "max."-Markierung liegen (Bild 3) und bei betriebswarmen Getriebe (Getriebeöltemperatur ca 80°C) muß bei richtigem Ölstand die Anzeige an der "max."-Markierung anliegen (Bild 3).	1	1
	15. Ölstand nochmals kontrollieren und ggf. berichtigen.	1	0

Figure 96: Benchmark Test Results (t_2) Setup 2 – Part 1

RESULTS RETRIEVAL BENCHMARK RECALL OF TEST CASES TASK 2 SETUP 2			
TEST CASE	TEST ITEM	RECALL SYSTEM 1	RECALL SYSTEM 2
change of sequence of main and sub-clauses	16. Ölstand im automatischen Getriebe richtigstellen, ggf. prüfen. 17. Bei größerem Ölverlust Ursache beseitigen und ermitteln. 18. Beide Teile entfernen und Verschlußhebel (6a) öffnen und Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken. 19. Ölmeßstab (6) herausziehen und bis zum Anschlag einstecken. 20. Sicherungsstift (6b) einsetzen, bis er einrastet, Verschlußhebel (6a) schließen.	1 1 1 1 1	1 1 1 0 1
change acronyms (only one available in test text)	21. ALP-Öl nach Betriebsstoff-Vorschriften Blatt 236.4/6/7	1	1
change numbers (n+1)	22. *Bild 2 links (bis 09/93)* 23. Bild 3 rechts (ab 10/93) 24. Verschlußhebel (7a) öffnen 25. Ölmeßstab (7) bis zum Anschlag einstecken, wieder herausziehen, Ölstand ablesen 26. Bei kaltem Getriebe (Getriebeöltemperatur ca. 31°C) muß bei richtigem Ölstand die Anzeige zwischen der "min." und "max."-Markierung liegen (Bild 4).	1 1 1 1 1	1 1 1 1 1
change date numbers (n+1)	27. *Bild 2 links (bis 10/94)* 28. Bild 3 rechts (ab 11/94) 29. bis 10/94 (Bild 1) 30. ab 11/94 (Bild 2) 31. bis 10/94 (Bild 1)	1 1 1 1 1	1 1 1 1 1
for formatted text strings: remove formatting	32. GETRIEBE 722.3/4/5 33. Bild 1 links (bis 09/93) 34. Bild 3 35. Prüfen 36. Richtigstellen	1 1 1 1 1	1 1 1 1 1
for formatted text strings: change formatting	37. GETRIEBE 722.3/4/5 38. **Bild 1 links (bis 09/93)** 39. **Bild 3** 40. *Prüfen* 41. *Richtigstellen*	1 1 1 1 1	1 1 1 1 1
for non-formatted text strings add formatting	42. **Ölstand** im automatischen **Getriebe** prüfen, ggf. richtigstellen. 43. Bei größerem Ölverlust Ursache ermitteln und beseitigen. 44. Sicherungsstift (6b) seitlich in **Pfeilrichtung** wegdrücken, beide Teile entfernen und **Verschlußhebel** (6a) öffnen. 45. Ölmeßstab (6) bis zum **Anschlag** einstecken wieder herausziehen, **Ölstand** ablesen. 46. *Verschlußhebel* (6a) schließen und Sicherungsstift (6b) einsetzen, bis er einrastet.	1 1 1 1 1	1 1 1 1 1
TOTAL RECALL OF TEST CASES		46/46	37/46
FUNCTION VALUE FOR RETRIEVAL BENCHMARK SETUP 2		100	80.43

Figure 97: Benchmark Test Results (t_2) Setup 2 – Part 2

In short, system x managed to retrieve all of the test items, while system 2 managed to retrieve 37 of 46 test items, resulting in:
v(x) = 100 v(y) = 80.43

Both setups were considered equally important to reveal the quality of the translation memory component for an industrial translation department such as that of Mercedes-Benz. Consequently, the calculation of the final value was performed by multiplying the function values of both setups with .5, leading to the following overall values for the quality of the translation memory retrieval component:
v(x) = 92.10 v(y) = 55.99.

The second type of benchmark applied for (t_2) measured *time behaviour*, that is, the seconds needed to retrieve translations from the translation memory. The test was performed with both setups and the results averaged. System 1 needed on average three times longer to retrieve translations (3 sec.) than system 2 (1 sec.). This was due to the time-consuming DDE-communication between WinWord and the TM application. In order to locate the values on the curve, it was decided that the function value decreases linearly, with a value of 5 seconds leading to a zero function value, resulting in:
v(x) = 40 v(y) = 80

4.4.4 Experiences with Scenario Testing

As pointed out before, the primary objective for performing a scenario test in the context of this thesis was to investigate the adequacy of scenario testing for evaluation preceding purchase decisions. A central question herewith was whether it is possible at all to arrive at reliable numerical values for metrics applied during scenario testing, which can be included into the assessment procedure as proposed in the evaluation framework. While other test types were presented only on a very basic level, the scenario test will be described in great detail here to allow the objective judgement of the complexity of this test type.

The following tables provide an overview of the problems tackled during the different phases of the scenario test.

PROBLEM	**TEST PLANNING PHASE**
costs	ca. 1 PM
software	system test with IBM TM/2 and Trados TWB for Windows
type of scenario test	field, integrated into Computer Aided Translation course
location	University of Helsinki; Kouvola 10 PC LAN/Windows
users	10 Finnish students with English Major or Minor
time	course: 15/05/95 - 19/05/95 explorative learning session: 22-23/05/95 pilot testing/observation session: 24/05/95 field test: 26/05/95
evaluators	M. Höge L. Carlson assistants: 10 students (participants of same CAT course)
PROBLEM	**TEST PREPARATION PHASE**
quality requirements	understandability, learnability, operability, suitability, fault tolerance; interoperability
test task	task 1. establishing translation environment; subtasks: • starting programs • opening test documents • opening translation memories • opening dictionaries 2. translating given text subtasks: • retrieving/storing translations • inserting terms from dictionaries • editing terms
test data	test corpora: manual of TM2 for Windows • text for training the TM • text for translation during test
instruments	• questionnaire to elicit user profile • scenario checklist • observation • post-testing interviews
equipment	• overhead projector • beamer
test plan: test procedure and duration	1. introduction 15/05/95 2. questionnaire 15/05/95 3. training course and. explorative learning session 15-19/05/95 22-23/05/95 4. pilot testing/observation session 24/05/95 5. field test 25/05/95 6. post-testing interview 25/05/95
PROBLEM	**TESTING PHASE**
organisation and time-management	tests: 26/05/95 9.45 - 11.15 user break/observer's discussion: 11.15 - 11.45 post-testing interview: 11.45 - 12.30
trouble shooting	problems: copying of test texts to local drives deviations from test plan: TM/2: 6 subjects, TWB: 2

Figure 98: Problems Tackled in Scenario Tests – Part 1

PROBLEM	DATA ANALYSIS PHASE
data viewing	analysing scenario checklists w.r.t. aspects related to pre-defined quality requirements
data collection	compare results of different subjects
calculation	value types: mostly qualitative
PROBLEM	**REPORTING PHASE**
documentation	document test-bed precisely document all decisions taken during all phases document all deviations from test plan provide total of testing data as appendix
evaluation	justify all interpretations of results summarise results

Figure 99: Central Problems of the Reporting Phase

As the above tables show, the scenario test was integrated into a Computer Aided Translation (CAT) course performed by Prof. Lauri Carlson and the author of this thesis for students of translation at the University of Helsinki. The systems taught and tested were Trados TWB for Windows ß (system x) and IBM Translation Manager 2 for Windows (system y). The objective behind testing was to assess the applicability of scenario tests for the purpose of evaluation preceding purchase decisions. The scenario test was split up into six major parts:

1. introduction
2. user profile questionnaire
3. training course and explorative learning session
4. pilot testing/observation session
5. field test
6. post-testing interview

In the following the major constraints and findings of the different parts will be presented, followed by a survey of scenario test results.

4.4.4.1 Introduction into the Problem of Scenario Testing

The CAT course started with a one hour introductory session in which students were presented with (i) administrative details of the CAT course, and (ii) technical details of translators' aids.

Administrative Details of the CAT Course

Students were told that the motivation behind the course was twofold, that is, to teach them two major CAT systems and to perform tests which were meant to elicit information on (i) the applicability of the testing framework developed in this thesis, and (ii) the quality of the two translation tools. For this purpose students were asked to fill in the user profile form, to take part in a training course of the two systems, to

perform explorative learning sessions, and to take part in a scenario test either as subjects or as observers.

Technical Details of Translators' Aids

Students were given an introduction into the technical details of translators' aids, since training theory points out that material should always be presented in a way that the larger context becomes clear. As Hohmann/Le-Hong (1993:10) point out, logical connections that are shown to the learner in advance will help him/her to remember the information presented. Consequently, the following description of translator's aids systems was given to the students at the beginning of the course: translators' aids normally consist of three major modules, that is,

(i) an editor in which a given text is translated
(ii) a dictionary/termbank module, from which translated terms are retrieved together with additional information such as grammar, context, definition etc., and
(iii) a translation memory, in which previous translations are stored and retrieved during the translation process.

Both editors and on-line dictionaries were theoretically well-known by the students. Thus, the major focus of both evaluation and testing was put on the functionality of a translation memory: In short, a translation memory works on the recognition of stored (SL/TL) segments that match a given input SL segment. Most translation memories store the (SL/TL) pairs in databases. The following figure illustrates the functionality of a translation memory program:

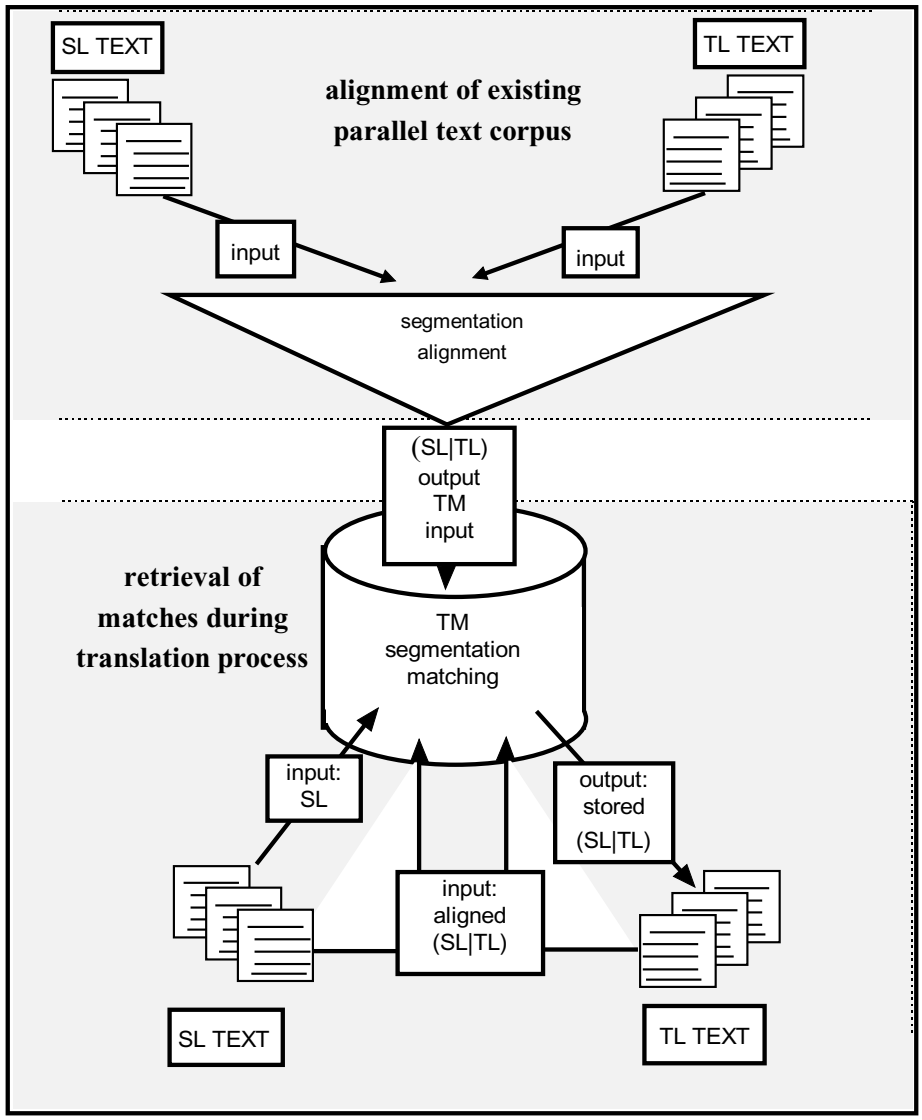


Figure 100: Technical Details of Translation Memories

There are two ways of building up translation memory databases, (i) by alignment programs, on the basis of existing translations or (ii) interactively, during the translation process:

(i) **Alignment programs**: An alignment program needs as input two separate files, that is, the SL text file and the corresponding TL text file. The program performs segmentation of both files on the basis of internal segmentation rules, which range from simple end-character recognition to linguistic procedures such as parsing. In a second step, the tool tries to match segments of the SL text file with segments of the TL text file. The quality of the alignment module is decisive for

the success of the whole procedure. In most cases the alignment process is based on statistical calculations, that are further supported by textual clues and, in some cases, formatting information. The result of the alignment phase, that is, a file with (SL/TL) pairs can finally be imported into the translation memory database.

(ii) **Interactively**: In most translation environments the translator is provided with a translation editor into which the SL text needs to be imported, displaying text and formatting tags of the SL text separately. In some cases, the translation memory program is appended to standard editing environments such as WinWord, which allows the original display of the formatted SL text. Most translation memory systems work on the basis of overwriting the original SL text with either a retrieved TL text segment or a manually inserted translation.

For the retrieval process, the system, when initiated by the user, performs automatic segmentation of the SL text, and tries to match the identified segment with those already stored in the translation memory database. There are two possible matches: (i) either the system can retrieve exactly the same SL text segment from the database, that is, finds a 100 % match, or (ii) the system retrieves a SL text segment from the database that bears a certain similarity with the original SL text segment, that is, offers a *fuzzy* match.

The quality of the retrieval module depends on two major factors: (i) the flexibility offered in defining variables for the retrieval of exact matches and (ii) the internal retrieval algorithm for fuzzy matches. The procedures for the retrieval of exact matches range from automatic recognition and substitution of dates and numbers to the pre-defined substitution of type names, brand names etc. The implementation of the internal retrieval algorithm for fuzzy matches is based on simple pattern matching procedures, statistical calculations, supported by linguistic clues and, in some cases, neuro networks.

Again, there are two possible situations after initiating the retrieval process. Either a match is presented by the system and the user decides that he/she will make use of at least part of the presented TL text in the original, or the system does not offer any match, in which case the user is prompted to fill in the corresponding translation. The finally acceptable (SL/TL) pair is then stored in the translation memory database and instantly available for further translation retrieval processes.

4.4.4.2 User Profile Questionnaire Survey

The assessment of the personal background of subjects was performed by the aid of a brief user profile questionnaire which was divided into three main areas covering student profile, translator profile, and experience with computers. The questionnaire can be found in appendix 4. While student and translator profile are only of general interest in a scenario test in which the translation quality is not considered, computer literacy, that is, the subjects' experience with computers, is decisive for the interpretation of test results.

Student Profile

Out of the 20 participants of the CAT course, those 9 students that had English as Major, and 1 who had English as minor subject were selected as subjects for the test. On average the subjects performed 3.7 years of English studies, with a variance between 2 and 6 years. As second or third subject they had Finnish, Swedish, German, Hungarian, French, Italian, Russian, Psychology, or Pedagogy. The remaining 10 students who functioned as observers during the tests had French, Swedish, Russian, Japanese, Korean, Spanish, or Maths as their major subject. Minor subjects were Spanish, German, Finnish Literature, Japanese, Estonian, Philosophy, or Pedagogy. All of them had at least school knowledge in English.

Translator Profile

The native language of 9 test subjects was Finnish, and of 1 subject Swedish. Of those who had Finnish as their native language 8 had English and 1 Swedish as second language. The one with Swedish as native language had Finnish as second language. Third languages were mainly Swedish, German, Italian, French, and Russian. Most students translate their first and second language into their native language, 6 of them additionally translate into the foreign language. The special fields covered natural sciences, law, technology, commerce, education, psychology, and TV translation. Among the 10 test subjects only 2 had professional translation experience: 1 student had 6 years of translation experience in industrial companies, translating technical texts such as manuals, specifications, process descriptions mostly from English or German into Finnish or Finnish into English; the other student translated over 1 year texts related to portrait painting from English into Finnish.

Among the 10 students who functioned as observers only 1 did not have Finnish but Estonian as native language. Second and third languages were French, Swedish, Russian, Japanese, Korean, Spanish. All observers primarily translate into their native

language, some additionally into the second and third languages. The special fields covered EU law, commerce, literature, history. Again only 2 students had previous professional experience which, however, ranged only from 3 days to 1 month translation practice in shipbuilding, architecture, and choir song texts.

Computer Literacy

Questions about the experience with computers were divided into four main areas: operating systems, word processing, CAT tools, and networking. For the interpretation of results the first three categories were of major interest. Around 70% of both subjects and observers proclaimed to have basic knowledge of the operating systems Windows and DOS, while Macintosh (25%) and UNIX (10%) were rather unknown. On the word processing side the profile was rather astonishing, since most subjects were only used to working with WordPerfect (9/10) while only 2 were used to working with WinWord. Among observers, the figure for WordPerfect was also 9/10, however, even more were used to working with WinWord (4/10). AmiPro was not used by students at all and the four other systems students referred to were Business Manager, PageMaker, TEKO, and a Japanese language kit. Among all students only 1 proclaimed to have regularly used a CAT tool, i.e. TRANSES 2.0, an electronic dictionary. No one else even tried out any CAT program.

The questions on networking revealed that both groups - subjects and observers - had unanimously little knowledge on networking aspects in general (22,5%), i.e. only few students were used to working with any of the networks available in Kouvola or Helsinki and only 5/20 had an own email account. For telnet, ftp, gopher and WWW the figures were even lower (on average less than 15 %).

To sum up, on the student and translation side, the profile of subjects displayed a good starting point for tests, though it would have been useful for subjects of user tests to have more professional experience. However, on the computer literacy side, the profile, which in general displayed low to medium computer literacy, already pointed to possible problems during both training and tests, since

(i) 6/20 students did **not** have any previous Windows experience, on which the handling of both systems is based

(ii) 8/10 subjects did **not** have any WinWord experience, on which the handling of the Trados system is based

4.4.4.3 Training Course

The training course was the most time-intensive part in the overall performance of the scenario test. The IBM software was taught primarily by Prof. Lauri Carlson, while the Trados software was taught by the author of this thesis. The original aim of the course was to train the 20 students in using both systems equally well before 10 of them perform scenario tests with the two systems. However, there is mostly a clash between training preparation and actual performance, as there often is a clash between theory and practice. In the following all considerations related to the preparation of the training course will be described, followed by the actual experiences made during the course.

Training Preparation

The following principles have to be considered when preparing training programmes:

- **Specificity of training programme:** Training methodologies cannot be taken from one subject and applied to another without considering the specifics of the subject. Moreover a training methodology has to be adapted to the final goal of training, that is, what should be the result of the training session? While in a "normal" training programme, the primary goal should be that users will be able to work with the system in their daily environment, the major focus of this training programme was to enable users to perform particular tasks with **two** systems in order to assess the quality of the **two** tools during a subsequent scenario test.
- **Avoidance of interference:** If similar material is trained in succession, interference occurs, which according to Birkenbihl (1990:pp.140) makes it harder to remember the information presented. Since the two systems trained are both geared to support the translator in his/her work, it is obvious that actions needed to be taken that decrease the problem of interference. The approach taken here is not to perform one task with system 1 and then immediately afterwards train the same task with system 2 (highest probability of interference), but rather to split up the group into two, one starting with TM/2, the other with TWB. After three hours (two sessions) of training with the first system, the trainee has learnt to perform the central tasks with the first system and then starts from scratch with the second. This procedure cannot guarantee that the problem of interference is totally eliminated, yet it leads to a decrease of the probability of interference.
- **Objectivity:** The concept of objectivity in training programmes only comes into play, if the systems trained are tested subsequently. If two systems offer similar functionality, users tend to judge the first system learnt more positively than the second, since the second system is not considered on its own right but in comparison to the first. The solution presented for avoiding the problem of

interference can at the same time help to increase objectivity during tests. When interpreting test results, particularly positive or negative responses from subjects can be traced back to the system which they learnt to use first.

- **Adequate level of difficulty:** The level of difficulty of the training sessions should be adjusted to the target group. If the group of learners, however, is not homogeneous, as could be drawn from the user profile questionnaire, special attention had to be paid to the problem of not asking too much from inexperienced users while at the same time keeping the interest of experienced users awake.
- **System vs. application orientation:** One of the most often cited problems in software training is the strong system orientation which can be found in training programmes elaborated and performed by software developers directly. If training is too strongly focused on the functionality of a specific tool rather than on the actual tasks performed by the user during the application of the tool in a given environment, the user might be at a loss when trying to apply the newly learnt procedures to the own application environment. Application oriented training programmes, such as the one developed in the context of this scenario test, do not take a specific function as the starting point of training, but start from a general task, and describe how a particular system has to be used to perform the given task. Since the training programme developed here integrates the training of two different systems, which offer different functions to perform particular tasks, the need for application orientation is even more obvious.
- **Investigation of user's task and function:** Application orientation in training programmes presupposes that the tasks be clearly defined. Therefore, the starting point for the development of each training methodology should be an investigation of the user's task and typical function within a company. The tasks selected for the training environment cover (t_1) - (t_4). Due to the limited amount of time available for each system (3 hours of training for each system) there was a clear focus on operative translation tasks.
- **Different channels for presentation:** Using different channels for the presentation of information, details can be remembered more easily. The training programme covered all three channels - listening, seeing, doing - to the maximum extent, verbal/on screen presentation of information as well as the actual usage of the system by trainees.
- **Explorative learning:** Users should be encouraged to explore the software on the basis of the knowledge already acquired. Proposals for exploring the system that are based on the knowledge gained during the performance of the previous tasks were given by the trainer. In addition to the explorative learning proposals, the framework also covered two 3 hours explorative learning blocks, in which trainees were asked to translate a given text by the aid of the two systems.

Actual Experiences of User Training

One of the elementary preconditions for successful training and testing is that both hard and software environment are well prepared. While the installation of TM/2 was performed on the LAN several weeks in advance, due to the late availability of the Trados system, the software could only be installed directly before the CAT course started. Thus the functioning of TM/2 could be assessed in due time before the course started, while it was found out only at the beginning of the course that the installation of the Trados ß-version on the LAN led to problems, mostly related to the fact that the installation routine was not followed exactly by the technical support personnel who installed the software, and, moreover, high hardware requirements could not always be met.[1] Also, WinWord was not installed properly on all machines before the course started, so that the preconditions for Trados to work properly at the first course session were rather bad.

As a reaction to the installation problems with Trados, the plan of splitting the groups into two, one starting with TM/2, the other with Trados could not be applied. Instead, all students first got to see TM/2, which shows that the principle of *objectivity* clearly could not be satisfied during the training sessions, since the conditions for TM/2 were much better than for Trados.

During the training of both TM/2 and Trados, it was found very difficult to find an *adequate level of difficulty*, since a number of students even had to be taught how to make use of the mouse and Windows. Thus for many students the learning capacity was exhausted after acquiring the necessary Windows skills and learning some of the systems' functionality. For Trados, the starting off was again much worse than for TM/2 since (i) when getting to see Trados, students did not have much learning capacity left, (ii) students learnt about the problems with Trados and thus feared that the system was not robust enough for explorative learning, (iii) in many cases insufficient hardware caused unacceptable response times, and, (iv) since only few of the students had previous knowledge of WinWord, they had to acquire the basic functions of WinWord before they could even think of using the Trados system. As a consequence to the above described problems, most students mainly used TM/2 during the explorative learning sessions. Only on the three machines which were powerful enough to run Trados successfully, more computer literate students actually performed explorative learning with Trados.

[1] The Trados system requirements as defined in the documentation were: (i) a PC AT with a 80486 DX or higher processor running at 33 MHz or more, with at least 8 MB of RAM. Recommended is a Pentium processor, 16 MB of RAM, and a 17" monitor; (ii) the following software environment: DOS 3.3 or higher, Windows 3.1 or Windows for Workgroups 3.11, Word for Windows 6.0

Due to the various problems encountered during the CAT course, it was decided at this point of the evaluation procedure that test conditions were not appropriate for the Trados system and thus mainly TM/2 could be assessed.

4.4.4.4 Pilot Testing/Observation Session

The success of direct observation depends to some extent on the experience of the observer. Thus a pilot testing/observation session was held which allowed the 10 totally inexperienced students to observe their counterparts and to fill in a scenario checklist. The students of the class received a 10 Min. introduction into the principles and goals of Scenario testing in general and of direct observation in particular. The scenario checklist designed for the test covered the following columns: test text, sub-tasks, functions, user comments, observation remarks, user errors/problems, help requests, system failure, time. The design of the checklist was supposed to allow measurement of the following metrics: *frequency of help/documentation use*; *understandability of user interface*; *frequency of user errors*; *suitability of fuzzy match proposals*.

During the pilot session, students were asked to observe the subjects until they feel fit in filling in the checklist. However, the maximum time for practising was set to 1 hour. Among the ten students performing the pilot observation, only two needed the maximum observation time. Major problems during the pilot observation were:

(i) identifying aspects that need to be noted as remarks
(ii) distinguishing between the concepts of sub-tasks and functions
(iii) watching and noting at the same time
(iv) identifying user errors

While problems (i) to (iii) could be resolved largely before the actual test started, problem (iv), that is, the identification of user errors remained to be a problem for most observers even during the scenario test. This was mainly due to the fact that, though being trained in using the software during the preceding 9 day training course, most observers did not feel fit enough to judge their companions' handling of the software.

4.4.4.5 Field Test

The scenario test displayed the typical characteristics of a field test, since it

(i) took place in the PC lab, in which students normally perform their computational work;

(ii) made use of the typical field test instruments, that is, direct observation, scenario checklist, post-testing interview, and
(iii) caused little expenses.

Observations as instruments deliver results related to the interaction between user and system. The type of results can be quantitative, e.g. the time needed to perform an action, the frequency of usage of functions, the frequency of failures or problems etc. or qualitative, e.g. the type of interaction chosen, the comments made by subjects, the type of problems etc. In a first step the observational data has to be collected from all subjects and thematically ordered. In a second step, aspects that need clarification have to be identified and discussed in a post-testing interview. Only after the collection of results by means of both observation and interviews, the final evaluation can take place, that is, the results can be related to ISO quality characteristics.

Test Proceedings

The testing environment was the same PC lab in which the CAT course was held. Among the 8 students available as subjects six chose to test TM/2 while two of the more computer literate students were prepared to test Trados TWB on the more powerful machines. As a help, students were allowed to make use of their notes and user manuals. In short, during a 90 minutes testing session, subjects were asked to translate as much as they can of a given English text into Finnish, making extensive use of the tools provided, while being observed by those students with whom they were associated during the preceding course. The scenario checklist was the same as the one used in the pilot testing/observation session with the test text displayed in the first column.

When testing two systems comparatively, it is very difficult to find a test text that is equally useful for both systems, that is, gives the same chance to both systems. For the testing of translators' aids, there are two elementary functional requirements on the test text: availability of (i) previous translations stored in the TM and (ii) terminology elaborated for the domain in the source and target languages. In this sense, the University of Helsinki identified the TM/2 for Windows manual as appropriate, since the manual of the OS/2 version of the very same system was previously translated at the University of Helsinki, making use of TM/2. Thus various translations were already stored in the TM/2 translation memory and corresponding terminology was elaborated and stored in a TM/2 dictionary. Again the test text was less appropriate for Trados, since the translations had to be aligned automatically by the Trados alignment tool TAlign, which leads to a much lower hit rate for translation retrieval than for manually translated texts, while at the same time also misalignments can occur that

influence system acceptance. The terminology provided by the University of Helsinki could be imported into the Trados Multiterm database without much preparation effort. However, apart from the preparation of both translation memory and terminology databases, also the format of the text is an important factor for the performance of the test. Unfortunately the text was only available in the original tagged IBM format, which caused acceptance problems with the Trados version during both alignment and retrieval.

The test task was restricted to SL text reception and translation. For the SL text reception part, subjects were asked to copy the text from a network to their local drive, starting the programs needed for translation, opening/importing the text into the systems, and selecting already existing translation memory databases and termbanks. During the translation process, subjects were asked to make extensive use of the translations provided in the TM, to look up parts of sentences in the translation memory databases, to look up and incorporate terms from the dictionary, and, where possible to edit (add/delete/change) dictionary entries.

Survey of Observation Results

Since it was decided after the training part that, due to various problems with the TWB training, a comparison between the two systems is not possible, the following presentation of results will focus on aspects related to the performance of TM/2. Where appropriate, short remarks will be given about comparable aspects of the performance of TWB4W. Among the many observations noted on the scenario checklist those will now be discussed that, after detailed examination of the matter, proved to be of importance to establishing a clear picture of the performance of the subjects relevant to the evaluation of the tools.

1. **SL Text Reception**: Observations showed that though being taught before, most subjects had problems with copying the test text from the network to their local drives. Particular problems were caused because, instead of copying the text, some students opened the file directly on the network drive or only moved it to the local directory. Thus the file was instantly either not accessible or even not available to other subjects, wanting to copy the file to the local drive. The problem could be resolved by the aid of support personell within 15 minutes. Despite clear descriptions on the notes, four of six TM/2 subjects had problems with importing the file into the translation environment. For some, the sequence of subtasks and steps necessary was unclear, others were not sure how to define the properties of the particular text. After successfully finishing the import subtask in TM/2, some students on less powerful machines, complained about the analysing process taking too long. Problems also occurred when, after a user

error, the size of the editor window was changed and the text went out of the screen, or the translation memory window disappeared and subjects did not know how to get back to normal. Opening the SL text file and establishing the translation environment within WinWord caused no problems for the two subjects testing Trados TWB.

2. **Translation**: For TM/2 the interaction during the translation process showed a clear preference of students for key combinations. The most frequently used combinations were those for storing the translation, deleting the remaining text of the segment, moving on to the next translation segment, choosing a match from those presented in the translation memory window, choosing a term from the list of terms presented in the dictionary window. The number of different key combinations used by subjects throughout the test varied from the three most important ones to seven more sophisticated functions. At most six different key combinations were used within one translation segment. The great number of different possibilities for assigning key combinations to functions led in many cases to user errors, because subjects mixed up which combination was assigned to which function. In such cases, subjects heavily complained about the difficulty to undo actions and return to previous states. During the observation one student complained about the confusing layout of the screen. The handling of the editor caused various problems, that is, moving words within the segment, returning to previous segments, unwanted deletion of tags, using the key combination for the deletion of the remaining text in the segment, when cursor was not at the end of the segment. In TM/2 100% matches were always inserted by means of the key combination. However, handling proved to be too complicated and time-consuming, if only parts of matches were appropriate for the particular context. Some subjects repeatedly chose to just read the proposal from the translation memory window and typing the appropriate part in manually. This was even more true for using the terms offered in the dictionary window: less than 10 % of all terms offered in the windows were actually inserted into the text by means of the key combinations. The strategy of finding the translations for terms that were not automatically displayed in the dictionary window differed largely between TM/2 and TWB subjects: while TM/2 subjects never opened the dictionary to look up related words, and thus either used a paper-dictionary or took over the English term in the target text, TWB subjects frequently browsed through the termbank or made use of the concordance facility to look for the unknown term. Despite some complaint about the correctness of terminology, only one of the subjects edited an entry of the dictionary once. The quantity of text translated during the 90

minutes test session varied from 18 to 70 segments. The variance could be traced back to the time-consuming handling of problems with importing the SL text (in one case it took 30 Min!), not making use of matches or terminology offered by the system, problems with understanding the SL text, user errors when mixing up key combinations. On average TM/2 subjects managed to translate 32 segments.

4.4.4.6 Post Testing Interview

The major objective of the post-testing interview in the context of this scenario test was to elicit more detailed information about the reasons for a particular behaviour during the test, and to clarify issues that could not be clearly defined by observational data alone. Therefore the form of a structured interview was chosen, in which the interviewer asks detailed questions which the interviewees one after the other have to answer. A group discussion seemed adequate, since no critical personal data was going to be elicited.

From a psychological view it proved to be ideal to start with general questions about likes and dislikes before eliciting detailed information about interaction and closing the discussion with subjective evaluation statements from users. Thus the questions can be split up into three major blocks: (i) general impression of software, (ii) detailed discussion of user behaviour, (iii) user evaluation statements.

(i) general impression of software

what did you particularly like?

All subjects liked the functionality of a translation memory and electronic dictionaries in general. Most subjects were quite impressed by the speed of translation retrieval in TM/2. Though only few made use of the function, all subjects liked the possibility to define shortcuts for most TM/2 functions. Though not being prompted to comment on Trados TWB, five of six TM/2 subjects as well as the two TWB subjects pointed out that they like the Trados look and feel, in particular colours, layout and icons (e.g. dictionary icon, national colours). One subject did not mind layout as long as functionality is OK. The Trados subjects were enthusiastic about both functionality and handling of the concordance facility and all TM/2 subjects would have liked something similarly effective in TM/2. All subjects noted positively that they realised a certain increase in speed during interaction with the systems.

what did you particularly dislike?

All subjects felt uncomfortable, thinking of the fact that there is no possibility of modifying stored translations in the TM/2 translation memory database. Most subjects

complained about the long process of establishing the translation environment for external documents. In particular the importing of files into TM/2 was found to be not straightforward and the analysis of the text took too long on the less powerful machines. While students that were not used to Windows programmes were mostly satisfied with layout and functionality of the editor, more computer literate subjects complained about the confusing layout of the text in the editor and the reduced functionality. In particular subjects complained about the format only being displayed in form of tags. Related to the restricted functionality of the editor, subjects complained about the uncomfortable modification of proposals from the translation memory or the dictionary and the lack of undo possibilities during the interaction with the editor.

(ii) detailed discussion of user behaviour

why did you prefer key combinations as primary form of interaction?

Two of six TM/2 subjects, were convinced that key combinations are the best solutions whatsoever, since, not being experienced with Windows/mouse, it is much quicker when typing in translations to use the keyboard for interaction than moving the hand to the mouse for direct manipulation. The remaining four TM/2 subjects pointed out, that in the TM/2 environment, that looks very similar to normal DOS applications, the user was not particularly motivated to use the mouse. In general they also found key combinations quite handy, except for the fact that, using so many combinations for different functions, the user tends to mix up different combinations, leading to user errors which are difficult to undo in TM/2. As possible solution they would prefer a mixture between key combinations, particularly for those functions that are performed while typing in translations, and additionally buttons or toolbars for a quick mouse click, instead of having to pull down the menu and selecting an option there.

how did you like the functionality and layout of the editor?

None of the subjects found the TM/2 translation editor particularly user friendly and easy to handle. Apart from the problems observed and noted during the test, subjects particularly complained about the fact that having used the overwrite mode for translation, it is impossible to verify or modify the translation after finishing a segment, since the original is not available. In this respect all subjects unpromptedly pointed out that the Trados solution is preferable, since the original SL segment is always visible.

how did you like the translation memory facility?

On the content side, subjects complained that for some passages only few translation proposals were available. TWB subjects then often succeeded with their translations,

making extensive use of the concordance facility, while TM/2 subjects manually translated the segments. Whenever matches retrieved from the translation memory could be used without modification (100% matches), subjects found the handling easy. Yet if only fuzzy matches were available, subjects found it more time-consuming to perform modifications with the editor than to type in the translation manually. The quality of fuzzy matches was generally judged acceptable.

how did you like the dictionary facility?
On the content side subjects again complained about the relatively low hit rate. All TM/2 subjects pointed out that, if possible, they would have used the concordance facility offered in TWB to look for terms.

why did you insert terminology rarely?
All subjects complained about the problems of automatically inserting Finnish terminology, since the effort of inflecting automatically inserted words is often higher than for manually typing in the inflected form.

why did you rarely edit (add/change/delete) terminology?
The reasons given for not having edited terminology were that (i) subjects did not really feel responsible for the terminology, (ii) subjects feared they could make some irreparable error to the database, and (iii) subjects found it too complicated to edit terminology.

(iii) user evaluation statements.
do you feel fit in using the software now?
Five of six subjects of TM/2 and both TWB subjects felt that the training was sufficient for performing operative translation tasks. One TM/2 subject would prefer to have more intensive training before using the system on his own.

would you use the software?
Two of six TM/2 subjects expressed wishes to use the software for their private translation exercises. Another two doubted that they would make use of the software, since they were not sure that translation quality and quantity can actually be increased using the software. The remaining two TM/2 subjects found it depending on the type of text to be translated. For repetitive texts, they would try to use it, while for less repetitive text types they would clearly not use it. Both TWB subjects said that they would like to use the software for future translation tasks.

would you buy it?
Given the restricted financial situation of the subjects, nobody would buy any of the two systems currently. Both TWB subjects expressed hopes that their future employer would buy the system.

4.4.4.7 Survey of Scenario Test Results

The overall procedure of this scenario test showed that quantitative measurement did rarely make sense. In the following, different metrics that were intended to be applied during the scenario tests will be discussed in terms of the problems related to their measurability in this particular context.

- *time needed for training programme*: the training programme covered 5 days of training and 2 days of explorative learning. However, due to the technical problems with Trados TWB4W, more time was spent with TM/2. Consequently measurement of hours spent with either system is not *valid*, since it is not possible to generalise from the circumstances of the experiment to the circumstances in real life.

- *time needed to achieve performance criterion*: computer literacy among subjects varied greatly. The measurement of time in relation to performance depends on whether or not a subject was used to the direct manipulation techniques offered by the systems. Consequently the measurement of time needed to achieve a performance criterion is not *reliable,* since other subjects with different computer literacy will need a different amount of time to achieve a performance criterion.

- *frequency of help/documentation use* and *frequency of user errors*: in order to arrive at a value relevant measurement of how frequently subjects were in need of help or made errors, one has to consider how quickly they performed their test. In other words, there was a great variance w.r.t. the amount of text that was translated during the text (between 18 and 70 segments) and consequently also the frequency that functions were being used by subjects. A calculation based on the average of segments translated and help needed/errors made considering this variance in translation speed and also computer literacy is not *value relevant* in this context.

- *fault tolerance with user errors*: since observers did not know the systems much better than the subjects they were observing, it was difficult for observers to judge, whether a certain behaviour is due to a user error or the system itself. Consequently hardly any user errors were noted on the observation lists. The measurement of errors is, therefore, not *valid*.

- *suitability of fuzzy match proposals*: though the observation checklist allows the counting of the number of times in which a fuzzy match proposal was used, the two systems did not have equal conditions, since the alignment between SL and TL segments in the TM/2 case was done during translation and in the TWB4Windows case by an alignment program. Consequently a counting of frequency of usage of proposals is not *valid* in this context.

- *response time for TM retrieval*: the hardware requirements were not equally met with both systems. Consequently with Trados TWB4W relying on DDE communication between Windows applications, on a PC which could hardly meet the system requirements. the response times were much worse than that of TM/2. The measurement of time for retrieval, therefore, is in this context not *valid*.

The above discussion showed, that the goal to arrive at quantifiable results for the above metrics could **not** be met in this particular scenario test. Nevertheless, a qualitative evaluation of the data arrived at by means of the profile, training experience, observation, and interview will be presented in the following tables. Due to the setup of the scenario test, the following assessment will focus on results related to the TM/2 software. Only where possible and relevant in relation to the evaluation of the TM/2 software, results of the restricted test with Trados TWB will be provided.

SUITABILITY			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
-	no editing of translation memory databases	+	editing of translation memory databases possible
-	reduced functionality of editor	+	full functionality of WinWord
-	no additional display of original SL segment	+	additional display of original SL segment
		+	concordance facility

Figure 101: Qualitative *Suitability* Evaluation

As the above figure shows, the *suitability* of TM/2 is judged less positively than of TWB. Subjects considered a number of functions as necessary that the test version of TM/2 did not have. However, it has to be kept in mind that (i) the above evaluation is based solely on the performance of operative translation tasks and thus provides only a partial view of the overall *suitability* and (ii) the results are based on the testing of version 1.0 of TM/2 and TWB ß.

INTEROPERABILITY			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
+	sharing of TM databases in network	+	sharing of TM databases in network
+	sharing of termbank in network	+	sharing of termbank in network
FAULT TOLERANCE			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
-	no undo after user error, e.g. using incorrect key combinations		
UNDERSTANDABILITY			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
-	importing procedure unclear		
-	editing process of terminology unclear		

Figure 102: Qualitative Evaluation of *Interoperability*, *Fault Tolerance* and *Understandability*

As to understandability, the concepts of translation environments including translation memories and terminology management systems were well understood by subjects. A certain experience with TM/2 will certainly decrease the importing difficulties, since users will then understand which information is needed in order to process files and which options the system offers to support the process.

LEARNABILTY			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
+	general: easy to learn for non-Windows literate users	-	general: requires Windows literate users
-	functionality of individual items difficult to remember because of DOS-like interface	+	functionality of individual items easy to remember because of sophisticated interface

Figure 103: Qualitative *Learnability* Evaluation

Though TM/2 is a Windows product, its similarity to DOS programs is high, which decreases the phobia of less computer literate users. However, the number of non-Windows literate users is steadily decreasing so that this positive aspect of TM/2 is gradually turning into a negative feature for future users, who expect a maximum of flexibility and usability.

OPERABILITY			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
+	possibility to assign key combinations to functions	-	only small number of pre-defined key combinations offered
-	no toolbars for quick mouse interaction	+	toolbars for quick mouse interaction
-	layout of translation environment confusing (no WYSIWYG in editor)	+	layout of translation environment clear (WYSIWYG, colours, icons, buttons etc.)
-	editing of translations in editor uncomfortable	+	WinWord editing facilities
-	uncomfortable modification of fuzzy matches in translation editor	+	-"-
-	long procedure for establishing translation environment for external documents	+	having once prepared WinWord template, opening of documents easy

Figure 104: Qualitative *Operability* Evaluation

The above figure shows that all negative features of TM/2 can be traced back to its usage of a separate translation editor which is not nearly as comfortable in using as WinWord, the current standard editor.

TIME BEHAVIOUR			
TM/2 1.0		TWB ß	
judgement	feature	judgement	feature
+	instant response for TM and dictionary retrieval due to internal processing of data	-	slow response times for TM retrieval on less powerful machines due to DDE communication
-	analysis process too long on less powerful machines	-	fuzzy search of terminology in Multiterm too long on less powerful machines

Figure 105: Qualitative *Time Behaviour* Evaluation

The scenario test showed clearly that both systems are geared to a more powerful hardware environment than was available at the test site. While time behaviour of TM/2, which does not need any external data communication processes, was still mostly acceptable, if not even remarkable, it was obvious from the very beginning of installation that system requirements could not even nearly be met for TWB, which needs power for the DDE communication and the processing of neural networks.

4.4.4.8 Conclusion to Scenario Testing

The above description of the procedure and results of the scenario test showed that despite a great deal of effort that was put into the preparation and performance of the test, the test could, as such, not be marked as successful in quantitative terms. The variables on the personal background side of subjects were too many and the problems

with hard- and software preparation too complex to be a sound basis for a reliable test. Moreover, the test showed that in tests involving users it is very difficult to arrive at *value relevant*, *valid*, and *reliable* quantifiable data. However, despite many deficiencies related to the performance of the test, a great amount of rich qualitative data could be obtained that depict the likes and dislikes of users in a realistic way.

In short, the tasks applied during scenario testing were (t_2)-(t_4), that is, mainly translation memory and termbank retrieval and updating termbanks. Among the 87 metrics only 13 metrics were applied in the scenario test. The metrics investigated the quality characteristics *suitability*, *interoperability*, *understandability*, *learnability*, *operability*, *time behaviour*, and *fault tolerance*. All of the metrics applied during scenario testing were also applied in another test type, mostly task-oriented testing or inspection. The scenario test, consequently, was a means to validate the results of the other tests.

4.5 Conclusion to Testing in Evaluation

The data in appendix 2 can be analysed as follows:

- among the 87 metrics defined, 59 (67%) proved to be *evaluation relevant* and will be input into the assessment calculation that will be demonstrated in the next chapter;
- 23 metrics (27%) were applied in more than one test type, leading to an overall figure of 110 applied metrics;
- the distribution of evaluation relevant metrics per test type is as follows:

TEST TYPE	number of metrics	of which are evaluation relevant	**not** evaluation relevant
inspection	35	19 (54%)	16
task-oriented testing	42	31 (73%)	11
interface-driven testing	16	13 (81%)	3
benchmark testing	5	5 (100%)	0
scenario testing	13	11 (78%)	2

Figure 106: Distribution of Evaluation Relevant Metrics per Test Type

- the distribution of scales applied during the different test types are as follows:

TEST TYPE	binary	binary nominal	ordinal	ratio
inspection	23	12	0	0
task-oriented testing	26	3	8	5
interface-driven testing	13	0	2	1
benchmark testing	4	0	0	1
scenario testing	6	0	5	2

Figure 107: Distribution of Scales Applied during the Different Test Types

- the distribution of scales among metrics applied was:

binary scale:	72	(64.9%)
binary nominal scale:	15	(13.5%)
ordinal scale:	15	(13.5%)
ratio scale:	9	(8.1%)

From the above figures it is possible to conclude that the evaluation results all in all can be considered **highly objective**, since 86.5% were measured on binary, binary nominal or ratio scales and only 13.5 % were measured on ordinal scales. Furthermore, 27% of all metrics, among them all metrics that were measured on ordinal scales, were validated by means of different test types.

Feature inspection proved to be an important starting point, since it covered 31.5 % of all metrics. Also, it proved to deliver totally **objective** results in form of only binary and binary nominal scales. At the same time, the results also show that *feature inspection* has to be supplemented by other test types, since only 54% of all metrics applied in *feature inspection* were *evaluation relevant*. This implies that it becomes important to find out **how** the features that are available are implemented.

The test type delivering the highest quantity of *evaluation relevant* results proved to be *task-oriented testing*. 37% of all metrics were applied in *task-oriented testing*, of which 73,8% proved to be *evaluation relevant*. Consequently the highest number of results that go into the final assessment calculation go back to *task-oriented testing*. The **objectivity** of results gained by task-oriented testing can also be rated high, since 80,9 % of results go back to binary, binary nominal and ratio scales. All of the ordinal scale results were validated by other test types.

Of the 16 metrics applied in *interface-driven testing* only 5, that is, those for *compliance* and *fault tolerance*, were not at the same time applied in some other test type. It follows that the major function of *interface-driven testing* in evaluation preceding purchase decisions is to validate results of other test types, mainly of *task-oriented testing*. There lies an important difference between evaluation preceding purchase decisions and evaluation supporting development where *interface-driven testing* is the major means to detect problems and inconsistencies.

Benchmark testing covered only 4.5 % of all metrics applied in the tests. As pointed out before, the reason for this lies in the problem of identifying units of systems that are measurable independently from user input. The **objectivity** of *benchmark test* results is rather high, measuring the metrics mostly on ratio or binary scales. All

metrics applied in *benchmark testing* proved to be *evaluation relevant*, which points to the fact that *benchmark testing* is very important to the assessment calculation, since it describes **how well** central functions of the software were implemented. The results of *benchmark testing* may very well turn into knock-out criteria in the assessment phase.

With 11,7 % of all metrics applied, *scenario testing* ranks rather low in terms of coverage. All of the metrics applied in the *scenario tests* were at the same time also applied in other test types. The fact that 78% of the metrics applied are *evaluation relevant*, points to the fact that *scenario testing* elucidates important aspects related to workflow, likes and dislikes, that can be used to validate the results of other test types.

To conclude, testing showed that in evaluation preceding purchase decisions, it is indeed promising to start off with *feature inspection* and supplement the results with, above all, *task-oriented* and *benchmark testing* that deliver important information on **how well** the features are implemented. *Interface-driven testing* plays a minor role in evaluation preceding purchase decisions and, therefore, can concentrate on measuring quality characteristics such as *compliance* and *fault tolerance*. The experiences made with *scenario testing* pointed to the fact that it is very difficult to arrive at *reliable*, *valid* and *value relevant* results. However, despite the problems of measurability, the scenario test should be used as final way to **validate** the results of other test types. This validation need not even be quantitative in nature, since qualitative notions as presented above also serve to validate the numerical results gained through other test types. Consequently, for evaluation preceding purchase decisions, the *scenario test* is considered as invaluable means of **validation** of results. Its role for evaluation preceding purchase decisions can, in its own right, be compared to the role of the *sensitivity analysis* performed in the decision analysis process. In other words, the scenario tests can show whether the figures obtained during the various stages of evaluation really make sense in the practical working environment concerned.

5. Assessment in Software Evaluation

In evaluation supporting the development process, assessment is more qualitative in nature. This is due to the fact that the interest behind evaluation is to improve the software rather than to measure its quality in quantitative terms. This is mirrored in the form of the result reports of the evaluation process performed in the TBW I and II projects; the author played an important role as evaluator of the two projects. Result reports covered *observations* rather than *metrics* and *proposals for modification* rather than *value functions*. The TWB projects can be used as model for the final assessment statement in evaluation supporting development. Höge/Hohmann/Le-Hong (1995, pp. 168-173) produced so-called *balance reports* which briefed the assets and shortcomings of each system under evaluation in qualitative terms, leading to a general appreciation of the systems, supplemented with the details of test results. Some results of TWB I and TWB II are described in appedix 1.

In the context of this thesis, the development of assessment procedures will concentrate on evaluating purchasing decisions, where assessment relies on the great number of values for metrics of evaluation and value relevant attributes that are the outcome of the testing phase. In the introductory chapter it was argued that assessment in evaluation closes the evaluation cycle by validating test results and relating them back to the users. Assessment procedures in decision analysis were presented for multiple attribute measurement (chapter 3.1.1). These procedures have to be applied and adapted to the problem of software testing and evaluation. In short, assessment in software evaluation involves the rating and combination of individual test results, arriving at a numerical representation of the adequacy of different software systems for a specific domain.

In section 5.1 methods and concepts for assessment will be discussed in the context of the evaluation of translators' aids' systems. In section 5.2 an approach to assessment in software evaluation will be elaborated that takes into account the problems presented. In section 5.3 this approach will be applied to assess the adequacy of the two translators' aids' systems under test, integrating the test results presented in appendix 2. We will show that quantitative assessment is possible for the evaluation of translators' aids' systems.

5.1 Quantitative Assessment and Translators' Aids

The ISO quality tree enumerates good properties of a software system, independent of the task. A value tree in decision analysis lists preferences of decision makers

concerning the problem at hand. In the context of this thesis it is important to investigate, to what extent a quality tree based on ISO 9126 can function as the basis for multiattribute utility measurement. A central issue, therefore, is to determine whether the condition of *value independence*, as required for the additive weighted model (cf. section 3.1.1.5.2), is satisfied for the ISO quality tree. It is interesting to note that while the quality models by Boehm et al. (1978) or McCall/Richards/Walters (1977) display various inter-relationships between quality characteristics, the ISO quality tree assumes no such relationships. Testing experience in the TWB I and TWB II projects, however, showed that there are various inter-relationships between different ISO 9126 quality characteristics:

Characteristic	related characteristic	nature of relationship
interoperability	efficiency	efficiency increases, if more resources can be assessed during the translation process (termbank, encyclopaedia, corpora, MT, etc.);
compliance	usability	usability increases if the user interface adheres to standards known to the user;
customisability	efficiency	efficiency of TMs increases, if product names, versions of products etc. can be entered as variables (higher match rates);
fault tolerance	usability	usability increases, the fewer steps are necessary to undo whatever fault has occurred;
recoverability	functionality	functionality increases, if, after a failure, the data is still available and correct;
	usability	usability increases, if the effort of recovering data is low;
operability	efficiency	efficiency increases, if the effort for operation control is reduced;

Figure 108: Inter-relationships between Quality Characteristics in Evaluation

The above table shows that the type of relationship between the different quality characteristics is strictly *monotone*, that is, an increase in one of the quality characteristics always means an increase in preference. Consequently, as stated before, that is, if the condition of monotonicity is satisfied for value dependent attributes, the additive weighted model can still be applied (cf. section 3.1.1.5.2).

It is possible to consider ISO 9126 a *generic* value tree for evaluating software systems and as such satisfies the condition of *value independence*, while objective tool properties may map on value in non-independent ways. In other words quality characteristics are needed to construct *independent* criteria on the basis of which assessment can be performed as weighted sum of the values of the criteria. Performing evaluation of translators' aids systems along the ISO quality tree as discussed above helps to develop system properties and guarantees and assessment function for those properties relevant to the specific evaluation context. Before developing metrics for the different properties identified by means of the quality tree, the evaluator has to

make sure that the detailed tree satisfies the conditions stated in decision analysis, that is, it should be

- complete
- meaningful
- assessable
- non-redundant
- manageable.

Another important aspect related to quality trees that function as value trees in assessment is that of structuring. While value trees in decision analysis were presented as closed, n-level deep structures, representing the preferences in a domain, a quality tree for evaluation does not necessarily have a finite number of levels but is rather an open-ended feature structure with an arbitrary number of ever-refined features. Consequently, in software evaluation, the depth of each branch of the tree may be varying. The following figure illustrates this problem in the case of the quality characteristics *efficiency* and *usability*.

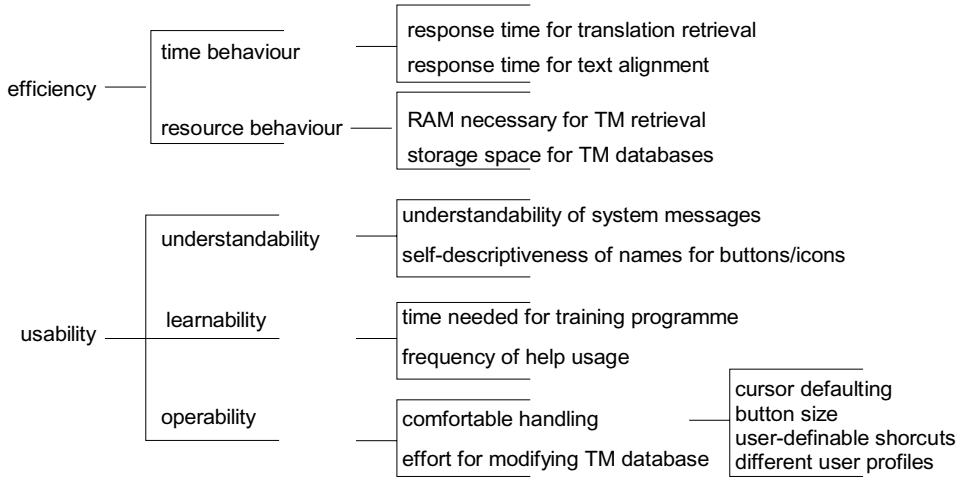


Figure 109: Quality Tree with Variations in Depth

The problem of varying depths in value trees is not satisfactorily discussed in decision theory. An assessment procedure for the evaluation of translators' aids should take into account the possibility of varying depths of branches when calculating the aggregate utility of an option.

The above quality tree moreover shows that one attribute may have a number of metrics which are measured on different scales. For instance the attribute *comfortable handling* is measured on

- binary scale: cursor defaulting? Y/N; user-definable shortcuts? Y/N; different user profiles? Y/N
- ordinal scale: button size: not acceptable -- optimal;

An assessment procedure for the evaluation of translators' aids has to discuss, how the ratings on the different scales are to be compared and combined.

In the current evaluation approach it is possible that metrics relevant to one and the same task may be applied in different types of test (scenario testing, systematic testing, feature inspection). Consequently, an evaluator may be confronted with different actual values for the same metric. An assessment procedure should consider the problems of combining different values for metrics performed during different test types.

In the software engineering context the so-called *actual values* (values arrived at by means of testing) are compared with *target values* that are defined in the software specification document and determine how exactly the system should behave. In other words, in white box testing, test suites are applied for which both test input and the expected test output (*target values*) are specified beforehand. In user-oriented evaluation of translators' aids' systems, the definition of *target values* involves the development of *value functions* that represent *preferences* over results. In other words, value functions are the central elements of the assessment procedure, since they map metrics onto quality criteria. They show the extent to which the relative value (utility) of a system increases with an increasing/decreasing value on the scale, where **0** means not acceptable and **100** means as well as one could hope to do. The evaluator has to find out whether more is always better or always worse than less (*monotonicity*) and whether the shape of the function is *concave*, *convex* or *linear*. Where monotonicity is not guaranteed, the evaluator has to determine saturation points, preference thresholds or peaks and develop value functions accordingly. The actual value then has to be located on the curve and the relative value determined.

A principal question in the context of developing value functions for evaluating translators' aids' systems is, whether there is but one possibility of assigning relative values to scale values for specific metrics. In other words, is a preference over test results necessarily valid for **all** tasks the system supports or may preferences change over tasks? Let us consider the following example: In interactive use, that is, during the translation process, the response time of a termbank has a clear saturation point: the quicker the response the better, until a certain point from where an increase in response time does not make any difference any longer, since it is no longer noticeable to the user of the system. When elaborating terminology lists of all terms in a text (as it

is used by interpreters, for instance) there is no saturation point, since the quicker each term retrieval, the quicker the overall response time for the terminology list.

The above example illustrates that in tool evaluation the range where linearity assumptions are valid, is restricted to specific tasks. Consequently, in the context of tool evaluation, value functions should be elaborated relative to the task being tested. An assessment procedure for the evaluation of translators' aids, consequently, should cater for the possibility of non-linearity in value functions, considering one and the same metric applied with different tasks.

Similarly, one may imagine that not only preferences with respect to individual metrics but also the importance of quality characteristics and metrics changes over tasks. In other words, while for batch applications, for instance, the quality characteristic *usability* is of marginal interest, for interactive applications, it may play a comparatively important role. An assessment procedure for the evaluation of translators' aids, consequently, should allow for the allocation of different weights for quality characteristics relative to the tasks performed.

Another interesting problem in the context of tool evaluation is that if value functions are task dependent, what happens to the assessment procedure in case of changing tasks? The elaboration of value functions is based on task requirements at *one given point in time*, that is, they mirror the "as is" situation. There is, however, no guarantee that the requirements will remain the same at *another point in time*. Software engineers frequently point out that the requirements of real systems are rarely static. Requirements change in response to changes in the environment. The introduction of software, for instance, leads to a drastic change of work flow, procedures, responsibilities etc., often resulting in a change of tasks. The problem of how to deal with changing requirements during the software life-cycle has gradually become a topic of interest in requirements engineering. There are some comprehensive approaches of how to deal with change such as by Chung/Nixon/Yu (1995) or Fickas/Feather (1995); which however, are geared towards sophisticated software development environments and therefore cannot be directly used for the purpose of evaluation. The complexity of identifying and monitoring change in the context of evaluation is immense. As a consequence, so far, it has not been dealt with at all in the context of evaluation. An Assessment procedure for the evaluation of translators' aids should consider the problem of changing tasks and a resulting change of both value functions and weighting during the assessment procedure.

5.2 Approaches to Assessment for Evaluation of Translators' Aids

The task plays a central role throughout evaluation. The approach taken during evaluation preparation, that is, weighting the importance of the different tasks in the domain, can be used and further developed in assessment by applying the *additive weighted model* on the task level. The following discussion will show, that this approach can cater for the various requirements of an assessment procedure as stated in the previous section.

When evaluating software systems, the assessment procedure has to be based on the task model elaborated during evaluation preparation. The procedure includes the following steps:

1. For each task relevant to the evaluation procedure develop quality tree considering quality characteristics, subcharacteristics and metrics.
2. Check validity of task quality tree. If necessary change tree accordingly.
3. Perform additive weighted model on the basis of individual tasks.
4. Perform *Pricing Out* procedure, relating the importance of tasks to the importance of costs and determine best match for specific evaluation context.

5.2.1 Developing Quality Tree for Evaluation Relevant Tasks

Weighting the importance of tasks during evaluation preparation in chapter 3, directed the effort that is put into the development of metrics. In other words, for important tasks more metrics are elaborated than for less important tasks. It has been demonstrated how to arrive at properties and metrics by applying qualitative aspects on tasks performed by the translator. In order to perform the *additive weighted model* on the task level, for each of the tasks $\{t_1,...t_n\}$, the quality characteristics $\{q_1,...,q_n\}$, subcharacteristics $\{sub_1,...sub_n\}$, and corresponding metrics $\{m_1,...m_n\}$ have to be defined and organised in form of a quality tree:

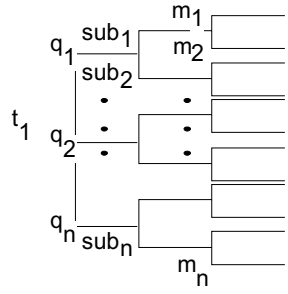


Figure 110: Quality Tree Relative to Task

A major issue when developing quality trees for the tasks relevant to evaluation is not to proliferate the branches of the tree. The "test of importance," advocated by Keeney/Raiffa for decision analysis, can also be applied in the context of tool evaluation. This implies that before any metric is developed to measure an attribute of the tree, the decision maker has to decide whether he feels his choice of a tool could be different if that attribute were excluded. Moreover, only if an attribute is *evaluation relevant*, that is, if systems under evaluation differ, it should be included in the tree, otherwise the evaluation process becomes unmanageably large.

5.2.2 Checking Validity of Quality Trees for Evaluation Relevant Tasks

Quality trees for individual tasks are based on *task models*. Validating quality trees for tasks, therefore, asks for reconsidering task models as they are developed in the requirements analysis stage. The following figure presents a possible process of validating task models in evaluation.

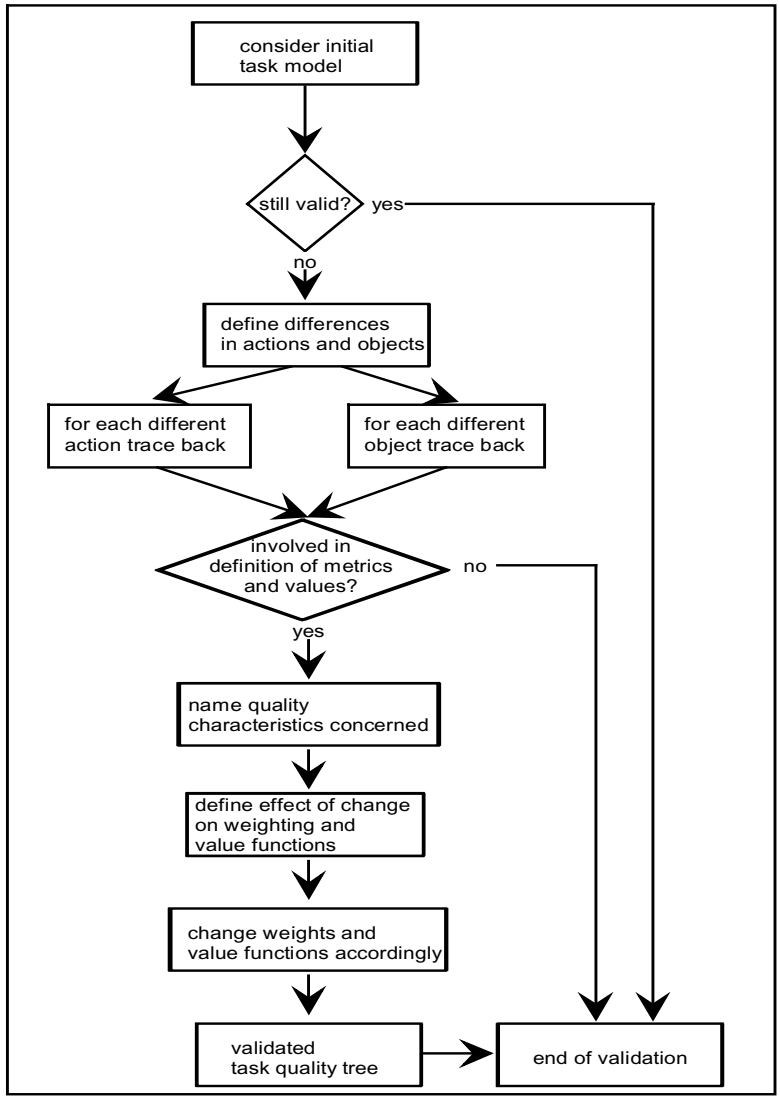


Figure 111: Procedure for Validating Task Quality Trees

Chung/Nixon/Yu (1995:pp.136) identify three major types of changes in requirements, that is,

(i) addition of a quality requirement
(ii) deletion of a quality requirement
(iii) changing the importance of a quality requirement.

According to Sewell (1990:287) each of these changes may have direct and indirect effects on the outcome of the assessment phase. The most obvious direct effect of the addition of a quality requirement is that either additional metrics are necessary to

check this new requirement or that the value functions for existing metrics have to be newly defined.

The deletion of a quality requirement may lead to the direct effect that either the tests and metrics used to check the requirement are no longer of interest for the evaluation result, or the value functions for existing metrics have to be newly defined. The direct effect of changing the importance of a quality requirement leads to a different weighting of the branches of the tree.

Indirect effects of changes in requirements are those that are concerned with *relationships* between quality characteristics. The best way to define the indirect effects of the changes of requirements, therefore, is to determine how the changes of one quality characteristic effect the related quality characteristics along the relationships identified and described before. The following table will illustrate this direct and indirect effects of changing requirements by means of an example:

initial requirement	during the translation process, an on-line termbank and an existing parallel text corpus should be consulted;
new requirement	during the translation process, an on-line termbank, an existing parallel text corpus, and a CD-ROM machine translation system should be consulted;
change of objects	initially two objects: on-line termbank, parallel text corpus; new: three objects: on-line termbank, parallel text corpus, CD-ROM MT system
quality characteristic	*interoperability*
direct effect	metrics for integration of CD-ROM results have to be developed; target: interface to the above three systems must be provided;
indirect effect	metrics for *suitability* of CD-ROM results have to be developed; target: higher quantity of translated text;

Figure 112: Examples for Changes in Requirements

5.2.3 Performing Additive Weighted Model on the Basis of Individual Tasks

The task-oriented approach to assessment asks for the application of the additive weighted model on the basis of individual task quality trees. As pointed out in section 5.1, however, the usage of quality trees for weighting may pose several problems that are not known in decision analysis. Thus, while many attributes that are three levels deep in the tree are directly measurable, others may require a further splitting up into smaller units that can be measured. The approach advocated here is that the *additive weighted model* should be applied no deeper than on the third level of the hierarchy, otherwise the evaluation procedure becomes too complex and difficult to follow for the decision maker, since to him higher branches of the tree may be more meaningful than the twigs.

A precondition for performing the *additive weighted model* on a three level deep task tree is that there must be values for all attributes on that level with which to perform further calculations. It follows that lower level attributes may have to be assessed and combined in a way that they together deliver a value representative for the higher level attribute.

A solution is that scale values of lower level attributes are mapped onto value functions, and the function values are then combined into an overall value depending on the weights of the individual attributes. To recall, when combining and comparing scale values the following issues have to be considered:

- The combination of values on nominal scales can be performed on the basis of transforming nominal scales into binary nominal scales, where **1** denotes the existence of a nominal value and **0** its non-existence. Whenever several nominal values are sub-ordinated to the same quality characteristic and bear the same weight the combination of binary nominal values can be performed by adding the numbers on the scale. Nominal values further have to be checked with respect to their dependence. If there are nominal values under the same quality characteristic that are mutually dependent, that is, one cannot do without the other, one zero value in the set leads to an overall zero. If there is one attribute of overriding importance in a set of n-values, any combination **not** including this attribute is also zero. Nominal values also have to be considered with respect to their importance**.** The weighting of individual features may lead to a situation in which enough minor good features can make up for a missing major good feature.
- For tool evaluation purposes ordinal scale values have to be mapped onto ordinal value functions and then combined. In the context of ordinal values, *Pareto optimality* implies that a solution which satisfies all criteria is best, a solution which satisfies no criteria is worst. In a strict mathematical sense, the rest cannot be compared. *Lexicographic preference* for ordinal scale values means that solutions better under the first criterion are better than others, solutions which are indifferent to the first criterion are ordered relative to the second and so on.
- Values on ratio scales can be combined and compared across systems, since concepts like "x is n-times as much as y" can be applied.

Another issue of interest in connection with lower level attributes is whether a lower level attribute can represent a threshold condition for the whole assessment, that is, failing the threshold to this attribute, the system fails the assessment on the whole. How should such threshold conditions be represented? At the moment, there is no mathematical representation at hand for this problem. One possibility would be to give the corresponding attribute, subcharactistic and quality characteristic a comparatively

high weight, so that in case of not meeting the threshold conditions, the effect on the aggregate utility would be tremendous. Future research could investigate other possibilities of representing threshold conditions of lower level attributes.

As further problem specific to user-oriented evaluation, the combination of results of one and the same metric arrived at by means of different test types has been identified. One of the major reasons for applying one metric with different test types is to check the reliability of test results. In other words, if a metric is classified as reliable, the results of one metric applied with the same task in different test types - for instance, examining termbank retrieval during translation in both scenario and systematic testing - have to be similar. If there is a strong discrepancy between the two values, the reliability of the tests has to be questioned and it has to be found out which of the results is representative for the situation, or even whether there is a flaw in the metric as such. If the results show only minor discrepancies, mean scale values can be taken for further calculations.

Taking into account the issues discussed above, the evaluator will arrive at values for all attributes three levels deep in the quality task tree. For each of these trees, weights have to be divided up among the branches and twigs of the tree as demonstrated in section 3.1.1.5.1. This involves the weighting of the importance of the different quality characteristics and attributes for the particular task, dividing 1 up among the attributes of each branch and multiplying these through the tree to arrive at the weights for individual metrics. In this way, the evaluator arrives at a unique figure of the aggregate utility of a particular system relative to a specific task.

This task-oriented evaluation approach provides detailed information as to how well different systems are capable of supporting different tasks. Since each task is performed by specific system functions, the decision maker can deduce the appropriateness of the different functions for specific tasks from the aggregate utility of the tasks.

To arrive at an evaluation score for each system, the aggregate task utilities have to be related to the pre-defined weights of the different tasks and the results added to form the utility of the overall system. The following figure illustrates how to arrive at the overall utility of a given system, based on the weights and utilities of individual tasks.

tasks	weights	aggregate task utilities	weight x aggregate task utilities
t_1	.35	45	15.75
t_2	.20	60	12.00
t_3	.45	60	27.00
total	1.00		54.75

Figure 113: Example for Calculation of Overall Utility Based on Task Utilities

To sum up, the task-oriented assessment approach demonstrated above is a combination between the quality-oriented approach advocated in software engineering and a problem-oriented approach advocated in decision analysis. It allows decision makers to determine the appropriateness of different systems for specific tasks and to rate the importance of the tasks for their specific environment. Differentiating between the performance of different tasks and the functions used to perform these tasks, the final choice of a decision maker may even ask for a combination of different functions offered by different systems under evaluation, for instance, making use of the alignment program of system (x) while using TM retrieval of system (y).

5.2.4 Utility and Cost - the Tradeoff Problem in User-oriented Evaluation

So far, the assessment procedure only considered software quality as criteria for assessment, that is, which of the systems or functions performs best for a particular task in a particular environment. In reality, decision making, however, also depends on the costs associated with a particular system. In the following, the methods used in decision analysis for relating utility to cost will be adapted to the task-oriented evaluation approach advocated for the evaluation of translators' aids' systems. This involves the following steps:

1. Relate system costs to aggregate utility of system
2. Perform *pricing out* procedure considering utility of tasks

The first step concerns the development of a value function determining the relative value of system costs for the specific evaluation environment. System costs in software evaluation includes

- purchase costs
- training costs
- maintenance
- update costs

The total of system costs associated with the choice of a specific system is then located on the curve of the value function as demonstrated in 3.1.1. This involves the choice of the type of curve used for the value function, that is, deciding whether the utility of the

system decreases proportional to an increase of cost (linear decreasing function), or whether the utility decreases not proportional to cost, that is, whether there are thresholds that cannot be passed, or whether the shape of the function is concave or convex. Locating the different overall costs of the systems on the curve, the evaluator arrives at a number between 0 and 100 representing the utility of a system with respect to its cost. For demonstration purposes let us assume the result of locating costs on a value curve lead to the following figures:
SYS1: 60
SYS2: 70
SYS3: 40
SYS4: 50

In a second step, the cost of the systems is then related to the utility of the tasks as elaborated during the assessment procedure. This approach differs from the one chosen in decision analysis, where the utility of costs is related to the utility of the different attributes in the quality tree and not related to the tasks. With the following sample table in which the utility of four systems with respect to three tasks is related to the utility of their costs, the decision maker gets an initial, general impression of the relationship between tasks and cost.

SYSTEMS	utility task 1	utility task 2	utility task 3	utility of costs
SYS 1	45	60	60	60
SYS 2	70	20	50	70
SYS 3	35	60	65	40
SYS 4	55	70	35	50

Figure 114: Example for Relating Task Utilities to Utility of Cost

If the decision maker does not want to perform any further calculations he/she could deduce from the above table that unless task 2 is of very high importance, the best match would be system 2, since it scores very high on task 1 and cost, and medium on task 3. The table also makes clear, what kind of tradeoff the decision maker has to made when choosing one system among the four. In other words, it becomes obvious where the decision maker has to give up something in order to follow his/her priorities.

If the decision maker wants to base his/her purchase decision on further calculations, the pricing out procedure can be applied at this level, in which the weights of the tasks as defined during the assessment procedure are related to the importance of the cost factor. For this purpose the cost factor has to be treated as one of the decision units of the assessment procedure, dividing 1.0 up among both tasks and cost. The following table shows how the weights were distributed "as old", that is, without considering costs as one of the decision units, and how the weights are distributed when cost is

considered one of the decision units, assuming that the "as old" importance relationship still holds:

UNITS	Distribution of weights "as old"	Distribution of weights in **pricing out** procedure
task 1:	.35	.24
task 2:	.20	.14
task 3:	.45	.32
cost:		.30
SUM	1.0	1.0

Figure 115: Redistribution of Weights in Pricing Out Procedure

The following table applies the pricing out procedure, leading to a final figure for different systems under evaluation, where

$V(t_{1-3})$ is the assessment value of task 1 to 3
$W_{t1-3}(V\ t_{1-3})$ is the weighted assessment value of task 1 to 3
$V(_c)$ is the utility of cost
$W_c(V_c)$ is the weighted utility of cost, and
V (X) is the evaluation score of systems x = 1 - 4

SYS	$V(t_1)$	$W_{t1}(V\ t_1)$	$V(t_2)$	$W_{t2}(V\ t_2)$	$V(t_3)$	$W_{t3}(V\ t_3)$	$V(_c)$	$W_c(V_c)$	V (X)
1	45	10.8	60	8.4	60	19.2	60	18.0	56.4
2	70	16.8	20	2.8	50	16.0	70	21.0	56.6
3	35	8.4	60	8.4	65	20.8	40	12.0	49.6
4	55	13.2	70	9.8	35	11.2	50	15	49.2

Figure 116: Example for Pricing Out Procedure Applied in User-oriented Evaluation

The above table shows, that the calculations support the above impression that system 2 is the best match for the evaluation problem at hand. It also shows that probably system 1 needs consideration as well since it performs almost equally well as system 2.

To conclude, the aim of this section was to demonstrate that a numerical representation of software evaluation results *is* possible on the basis of a common decision theoretic utility model. So far, evaluation models for translators' aids' systems have rested mainly on qualitative assessment statements of system quality and adequacy. They did not allow the comparison of different systems on numerical terms.

The task-oriented assessment approach takes up software quality characteristics and applies them on the level of individual tasks. Splitting the assessment procedure up into individual tasks, the condition of *value independence* is satisfied. Also, within one and the same task there is no possibility of a non-linearity of value functions as it

could be, if the assessment procedure was not task-oriented. From a practical evaluation angle, the task-oriented assessment approach allows the decision maker to get more problem-oriented evaluation information. In other words, it becomes obvious how different systems perform with respect to different tasks and shows where tradeoffs have to be made in order to find the best match for the particular evaluation environment.

In the next section the task-oriented assessment procedure advocated above will be applied with the practical test results presented in appendix 2. It will show that the different procedures and mathematical prerequisites are able to cover this type of evaluation situation.

5.3 Applying the Quantitative Assessment Procedure in a Practical Context

In evaluation preceding purchase decisions, the quantitative approach to assessment, developed in this thesis is invaluable since it will inevitably lead to a utility value for each system under evaluation, allowing their comparison. The following quantitative assessment example is based on (i) the needs as they would be valid for of a large translation department such as the translation department of Mercedes-Benz, and (ii) the task trees and practical test results presented in appendix 2. To recall, the tasks that were subject to testing were:

t_1 translation memory preparation
t_2 translation memory and termbank retrieval
t_3 updating translation memory databases
t_4 updating termbanks

5.3.1 Calculating Aggregate Utilities

The procedure for assessment presented below will, in principle, follow the approach presented in 5.2. For ease of presentation, the development of task trees and calculation of task utilities will be presented in the same figure, covering the results of the following five steps:

1. Since only *evaluation relevant* metrics will be part of the final assessment calculation, the first step in this procedure is to analyse the test results presented in appendix 2 and select those metrics for each task that proved to be evaluation relevant. Together all evaluation relevant metrics related to different quality characteristics constitute the quality tree that is relevant to the specific task.
2. In order to define the importance of the individual subcharacteristics and metrics, weights have to be distributed among the different branches of the tree

by dividing 1 up among all quality subcharacteristics w(s) and dividing 1 up among all metrics w(m) belonging to one subcharacteristic.

3. In order to arrive at the final importance of each metric, the weights have to be multiplied through the tree, arriving at w(i), that is, the weight of the individual metric.
4. Multiplying the weight of each metric w_i with the value of each system for the particular metric, one arrives at the relative values $v_i(x_i)$ and $v_i(y_i)$.
5. Summing up all $v_i(x_i)$ and $v_i(y_i)$ separately, one arrives at a utility figure of system x and y for each task.

The following figure represents the steps 1-5 for ($\mathbf{t_1}$):

w(s)	sub characteristic	w(m)	metric No	w(i)	v(x)	$v_i(x_i)$	v(y)	$v_i(y_i)$
.80	suitability	.03	m 3	.024	0	00	100	2.4
		.05	m 8	.04	100	4	80	3.2
		.05	m 9	.04	100	4	0	00
		.02	m 11	.016	90.20	1.44	90.43	1.44
		.1	m 12	.08	92.57	7.4	86.00	6.88
		.75	m 13	.6	83.43	50.05	78.82	47.29
.10	compliance	.5	m 14	.05	95	4.75	90	4.5
		.5	m15	.05	90	4.5	100	5
.05	customisability	1.0	m 16	.05	100	5	0	0
.05	installability	.5	m 17	.025	0	00	100	2.5
		.5	m 18	.025	0	00	100	2.5
1.0		4.0		1.0		81.14		75.71

Figure 117: Results of Applying Additive Weighted Model on Task 1

According to the result report in appendix 2, for ($\mathbf{t_1}$), 11 metrics proved to be evaluation relevant. Though the actual test results were often different, the weighting process for ($\mathbf{t_1}$) led to utility figures for both systems that were not far apart with

v(x) = 81.14 v(y) = 75.71

w(s)	sub characteristic	w(m)	metric No.	w(i)	v(x)	$v_i(x_i)$	v(y)	$v_i(y_i)$
.60	suitability	.02	m 21	.012	100	1.2	80	0.96
		.045	m 24	.027	100	2.7	0	00
		.025	m 25	.015	50	0.75	0	00
		.02	m 26	.012	100	1.2	80	1.2
		.1	m 27	.06	100	6	0	00
		.01	m 29	.006	100	0.6	0	00
		.01	m 30	.006	100	0.6	0	00
		.01	m 31	.006	100	0.6	0	00
		.4	m 32	.24	92.10	22.10	55.99	13.43
		.05	m 33	.03	100	3	0	00
		.2	m 34	.12	90	10.8	70	8.4
		.025	m 35	.015	85.7	1.3	28,6	0.43
		.01	m 36	.006	100	0.6	0	00
		.075	m 37	.045	0	00	100	4.5
.05	compliance	.6	m 41	.03	100	3	0	00
		.3	m 42	.015	100	1.5	0	00
		.1	m 43	.005	100	0.5	0	00
.01	security	1	m 44	.01	100	1	0	00
.09	customisability	.2	m 45	.018	100	1.8	0	00
		.3	m 46	.027	100	2.7	0	00
		.5	m 48	.045	100	4.5	0	00
..015	fault tolerance	.6	m 49	.009	100	0.9	0	00
		.15	m 50	.00225	100	0.225	0	00
		.25	m 51	.00375	100	0.375	0	00
.055	understandability	.6	m 52	.033	100	3.3	40	1.32
		.25	m 53	.01375	100	1.375	60	0.825
		.15	m 54	.00825	100	0.825	40	0.33
.1	operability	.15	m 56	.015	100	1.5	0	00
		.1	m 60	.01	0	00	100	1.0
		.15	m 61	.015	100	1.5	0	00
		.3	m 63	.03	100	3	60	1.8
		.3	m 64	.03	100	3	40	1.2
.05	time behaviour	1	m 65	.05	40	2	80	4
.015	testability	1	m 67	.015	80	1.2	40	0.6
.015	installability	1	m 69	.015	50	0.75	16,7	0.25
1.0		10.00		1.0		86.4		40.24

Figure 118: Results of Applying Additive Weighted Model on Task 2

The fact that for ($\mathbf{t_2}$) 36 metrics proved to be *evaluation relevant* shows, that ($\mathbf{t_2}$) was of central importance for this evaluation procedure. The utility figures of ($\mathbf{t_2}$) show a great difference between the two systems, with

v(x) = 86.4 v(y) = 40.24

A major factor in this calculation was the relatively high weight that was given to the benchmark test, measuring the quality of the TM retrieval component (metric 32).

w(s)	sub characteristic	w(m)	metric No.	w(i)	v(x)	$v_i(x_i)$	v(y)	$v_i(y_i)$
.4	suitability	.7	m 71	.28	100	28	0	00
		.3	m 74	.12	100	12	0	00
.1	security	1	m 75	.1	100	10	0	00
.1	fault tolerance	1	m 76	.1	100	10	0	00
.075	understandability	1	m 77	.075	80	6	0	00
.075	learnability	1	m 78	.075	83,3	6.25	0	00
.25	operability	.6	m 79	.15	100	15	0	00
		.4	m 80	.1	50	5	0	00
1.0		6		1		92.25		00

Figure 119: Results of Applying Additive Weighted Model Task 3

For ($\mathbf{t_3}$) **,** that is, updating translation memory databases, 8 metrics proved to be *evaluation relevant*. As the following table shows, system x (Trados TWB4W) performed very well for task 3, while system y (IBM TM/2) did, at the time, not provide any function for modifying the database during or after the translation process:
v(x) = 92.25 v(y) = 0

w(s)	sub characteristic	w(m)	metric No.	w(i)	v(x)	$v_i(x_i)$	v(y)	$v_i(y_i)$
.4	suitability	1	m 82	.4	0	00	100	40
.35	security	1	m 83	.35	100	35	0	00
.2	understandability	1	m 85	.2	80	16	60	12
.05	learnability	1	m 86	.05	73,4	3.67	66,7	3.33
1.0		4.0		1.0		54.67		55.33

Figure 120: Results of Applying Additive Weighted Model Task 4

Only 4 of the metrics applied during the tests proved to be *evaluation relevant to* ($\mathbf{t_4}$) , that is, updating termbanks. The performance of both systems with respect to ($\mathbf{t_4}$) is again rather similar:
v(x) = 54.67 v(y) = 55.33

The above four tables depicted the performance of systems x and y with respect to the four tasks under testing. The task-oriented evaluation and assessment approach asks for the weighting of the tasks during evaluation preparation as demonstrated in section 3.2.1.1. This task weighting can now be used as the basis for the calculation of the overall utility of the two systems as a sixth step in the assessment procedure:

tasks	$w(t_i)$	v(x)	$w(t_i)$ v(x)	v(y)	$w(t_i)$ v(y)
t_1	.3	81.14	24.34	75.71	22.71
t_2	.45	86.4	38.88	40.24	18.10
t_3	.125	92.25	11.53	00	00
t_4	.125	54.67	6.83	55.33	6.92
total	1.00		71.58		47.73

Figure 121: Overall Utility of Both Systems Based on Task Utilities

The above table shows how the two systems perform in total, not considering the aspect of cost. With an overall evaluation score of

v(x) = 71.58 v(y) = 47.73

system x (Trados TWB4W) is the clear winner in terms of adequacy for the specific environment.

5.3.2 Pricing Out Procedure for Systems X and Y

The cost factor is usually omnipresent in the context of evaluation preceding purchase decisions. Despite the rather clear outcome of the evaluation procedure in qualitative terms, the pricing out procedure will provide an even clearer picture as to which system to prefer.

The first step of the pricing out procedure is to develop a value function determining the relative value of system costs. The costs listed go back to the Trados price list of 06/03/95 and IBM information of the same time. Multiple licence costs will not be considered here, since that is mostly a matter of negotiation.

System 1:	TAlign:	5.600 DM
	TWB4W:	4.800 DM
System 2:	TM/2 (including ITM):	2.149 $ (at the time 1,0 $ ≅ 1.72 DM)
	in DM.:	3.700 DM

Let us assume that in the translation environment of the example one person would be responsible for ($\mathbf{t_1}$) , that is the preparation of translation memory databases, while ($\mathbf{t_{2-4}}$) would be performed by 40 translators. It has to be noted that support and maintenance could not be considered in this example, since this is a matter of contract and data was not available for the example. Discounts were available for both systems. For 40 licences the discount was approximately 50%, leading to the following total costs for both systems:

System 1:	1 x	TAlign:	5.600 DM
	40 x	TWB4W:	94.400 DM
	total cost:		100.000 DM
System 2:	40 x TM/2 (including ITM):		74.000 DM

Assuming that the preference of the systems decreases linearly with an increase in cost and the upper price limit is 150.000 DM, the relative utility of costs is:

Cost utility of System x: v(x)=37.06

Cost utility of System y: v(y)=50.66

The next step is to relate the task utilities to the utility of cost as demonstrated in the framework chapter. The following table provides an overview of cost utility and task utilities prior to weighting.

SYSTEMS	utility (t_1)	utility (t_2)	utility (t_3)	utility (t_4)	utility of costs
SYS x	81.14	86.40	92.25	54.67	37.06
SYS y	75.71	40.24	00	55.33	50.66

Figure 122: Sample for Relating Task Utilities to Utility of Cost

The above table shows what kind of **tradeoff** had to be made in the example. For a higher utility of system x for (t_1): 5.43, (t_2): 46.16 and (t_3): 92.25, one had to give up 0.63 points for (t_4) and 13.6 points in **cost utility**. Since the results in the example are very obvious, the assessment procedure could have stopped at this point in favour of system x. For the sake of completeness the *pricing out* procedure was performed and the importance of the money factor was related to the importance of the different tasks, delivering final utility values for both systems. For this purpose the cost factor was treated as **one of the decision units** of the assessment procedure, dividing 1.0 up among both tasks and cost. The following table shows how the weights were distributed "as old", that is, without considering costs as one of the decision units, and how the weights were distributed when cost was considered one of the decision units, assuming that the "as old" importance relationship still holds:

UNITS	Distribution of weights "as old"	Distribution of weights in **pricing out** procedure
task 1:	.30	.25
task 2:	.45	.40
task 3:	.125	.12
task 4:	.125	.12
cost:		.11
SUM	1.0	1.0

Figure 123: Redistribution of Weights in Pricing Out Procedure

The following table applies the pricing out procedure, leading to a final figure for the two systems under evaluation.

	w	v(x)	w v(x)	v(y)	w v(y)
task 1:	.25	81.14	20.28	75.71	18.92
task 2:	.40	86.40	34.56	40.24	16.09
task 3:	.12	92.25	11.05	00	00
task 4:	.12	54.67	6.56	55.33	6.67
cost:	.11	37.06	4.07	50.66	5.57
SUM	1.0		76.52		47.25

Figure 124: Evaluation Scores of System x and y after Pricing Out Procedure

The assessment procedure applied above showed that system x (Trados TWB4W) is the clear winner over system y (IBM TM/2) also after the pricing out procedure with
V(x) = 76.52 v(y) = 47.25

Factors responsible for this clear outcome were

- missing functionality of system y for ($\mathbf{t_3}$) and retrieval benchmark test (m 32) functioned as knock-out criteria;
- great difference in quality of TM retrieval component;
- great difference in overall look and feel.

5.3.3 Experiences with the Assessment Approach

The experiences made throughout the assessment procedures were manifold. First of all, the task approach advocated and applied in this thesis was essential to keep the weighting procedure manageable. In other words, it would have been impossible to put the many *evaluation relevant* attributes into the same value tree. The 36 metrics relevant to **task 2** presented the upper limit of size of a manageable value tree. The weighting process increases in difficulty with the number of metrics relevant to a task. Incorrect weighting, however, might disturb the reliability of the whole evaluation process. Consequently, a check routine was included to make sure that the weighting process truly reflects the decision maker's preferences:

> For each new calculation of w(i), compare w(i) with similar weights of other metrics in the same and/or other task value trees. If weights do **not** reflect preferences, w(s) and w(m) have to be reconsidered, newly calculated and re-checked until weights truly reflect the decision maker's preferences.

As result of the above described check routine, the weights in the value tree for **task 2** were reconsidered six times during the weighting procedure.

Furthermore it was found that the more metrics are subsumed under a quality subcharacteristics the higher w(s) has to be in order to truly reflect preferences. In contrast, whenever there is only one or few metrics per quality subcharacteristic, w(s) needs to be low otherwise w(i) is overrated. The following example demonstrates this:

> Task 1: *customisability* subsumes only 1 metric.
> Initial weighting: if w(customisability) =.1 and w(m) = 1 then w (i) = .1. Reconsideration: w(customisability) = .05 multiplied with w(m) = 1 leads to w(i) = .05, which better reflected preferences.

Also, it was found that metrics that deliver boolean values generally have to be weighted lower, since the discrepancy between the two possible values of 100 or 00 have an extreme effect on the final utility value.

The evaluation procedure showed that most functions were linear and thus validated the corresponding view held by decision makers. There were no threshold conditions that one of the systems did not meet. High weighting of individual metrics, such as for the TM retrieval benchmark, may function as knock-out criteria. In the specific assessment procedure, the combination of values proved to be no problem, since all values were considered independent from others. The combination of results of different test types proved to be no problem either, since the values were exactly the same in all test types, which in turn proved the reliability of the tests.

Summary and Conclusion

The motivation behind developing and refining an evaluation framework over more than a decade is rooted in the lack of methods for evaluation as experienced with the evaluation of the different TWB modules in the ESPRIT projects (Kugler/Ahmad/Thurmair, 1995). Since then the objective has been to develop a framework that could be applied in software development projects, but even more importantly nowadays, in the system purchasing context. The experiences of many different existing evaluations performed with translators' aids and other software systems found their way into the framework. Two typical *evaluation situations* were distinguished, that is, evaluation preceding purchase decisions and evaluation supporting the development process. Depending on the evaluation situation, the difference in steps involved in the evaluation procedure was described. The framework is supposed to help consultants or even user-organisations to facilitate the decision process, that is, which of the available translation systems should be purchased in their particular situation? Or, in the context of software development projects, the framework will lead to an improvement of the systems.

Applying the framework in a practical context showed that it is promising to follow an interdisciplinary approach to the problem of evaluating translators' aids' systems. The framework integrates findings from translation theory and practice as the baseline representing the application area, and is based on existing evaluation research in this field. Dealing with software as the object of evaluation, many ideas from software and requirements engineering could be usefully applied when defining what users want and how this can be tested. Last but not least, decision analysis provided the measurement theoretical basis for the quantitative assessment of individual attributes and their combination in a procedure that allows the quantitative comparison of the adequacy of different systems for a specific domain.

In short, evaluation is considered as cycle, starting off with eliciting and describing features of a domain {D}, concentrating on the tasks performed in the domain; and machines {M}, covering the functions to perform given tasks. Modelling, as next step in the preparation of the evaluation process, serves to structure domain and machine information. The outcome of the modelling phase are metrics and their corresponding scales, measuring the attributes relevant in the domain, as well as the definition of value functions for each metric, showing how different scale values determine the desired quality of the attributes. By means of three test types, that is, feature inspection, systematic testing and scenario testing, values can be obtained for all

metrics. Assessment provides mechanisms to combine individual test results into an overall evaluation score, which reflects the preferences of users in the domain.

The thesis was structured in accordance with the evaluation cycle, starting off with discussing the context of the application, that is, translation theory and practice, as well as major evaluation initiatives in the first chapter. The translation context showed, that in a time of growing internationalisation, the problem of documentation and its translation must not be underestimated. The discussion of the translation process, specifically the looping model developed by Nord, provided the basis for the elaboration of tasks and strategies used for problem solving during the translation process. The professional contexts in which translators nowadays work were described and practical problems related to the translation process identified. Translation as a problem solving process was considered from the angle of cognitive psychology, leading to the description of the type of knowledge structure applied during the problem solving process. An attempt was made to consider whether epistemic or heuristic techniques form the basis for various translation strategies used during text analysis, transfer and synthesis. Taking into account theoretical considerations and practical problems, candidates for automation were identified, including modules that (i) identify repetitions and retrieve already translated texts; (ii) offer the retrieval of encyclopaedic information; (iii) provide access to mono- and bilingual text sources; (iv) provide terminological information for source and target languages; and (v) assist in the elaboration of terminological information. It was argued that these features would help to guarantee the competitiveness of the translation process in the electronic age and, therefore, their availability and quality in computer aided translation systems must be evaluated. Apart from translation issues, the first chapter provided some insight into the problem of evaluation and the world-wide initiatives that have dealt with this problem at large. It was explained in which respect the framework developed in this thesis differs from and can add to already existing approaches.

Though translation theory and practice could deliver an understanding of what type of features would be useful when transferring a document from one language into the other, it could not provide any mechanism to describe these features in terms of detailed functional requirements. In chapter 2, requirements elicitation procedures stemming from requirements engineering were presented. Jackson/Zave's model for requirements formulation could be applied to the problem of evaluation and formed the basis for a new model of evaluation, in which the intersection between domain and machine attributes constitutes the evaluation space. Considering a number of requirements' studies performed in the context of translation, different dimensions and parameters were identified that allow the structured and detailed elicitation of domain

properties relevant in the translation context. For each dimension, the basic elicitation techniques stemming from knowledge engineering were presented, and an exemplar task description as outcome of the elicitation process provided.

Evaluation preparation is a complex matter and has so far not been dealt with satisfactorily. Chapter 3 provides detailed information on how the disciplines of decision analysis and software engineering deal with the structuring of evaluation relevant information. The approaches presented by these disciplines were applied to the problem of evaluation preparation. The structuring techniques used in decision analysis and software engineering were investigated. It was found that ISO 9126 can be considered a generic value tree that is needed to construct independent criteria on the basis of which assessment can be performed as the weighted sum of the values of the criteria. Consequently, performing evaluation of translators' aids' systems along the ISO quality tree helps to develop system properties and guarantees an assessment function for those properties relevant to the specific evaluation context. Moreover, the categorisation and generalisation principles from requirements modelling could be applied in evaluation to bridge the gap between user requirements and metrics. Scale construction issues in decision analysis provided the basic measurement theoretic principles for metrics, which map test results on binary, nominal, ordinal or ratio scales. Value functions used in decision analysis provided the mathematical prerequisites needed to relate possible to desired scale values for each metric and thus showed how acceptance levels for user requirements can be defined. By means of many examples it was demonstrated which modelling procedures can be applied in the two evaluation situations in order to bring together the different types of information that are relevant to elaborate metrics, that is, domain information, system information, qualitative information and test information.

In chapter 4 testing principles were investigated. Both glass and black box testing approaches applied in software engineering influenced the development of user-oriented test types for the evaluation of translators' aids. A goal-oriented model of test types was developed that demonstrates how values can be obtained for metrics in user-oriented testing. Scenario testing, which takes up acceptance testing principles from software engineering, was defined as a means to assess the appropriateness of a piece of software for every-day work. Systematic testing was defined to examine the behaviour of software under specific conditions. It includes approaches from software engineering that led to the distinction between three types of systematic tests, that is, task-oriented testing, interface-driven testing, and benchmark testing. As third test type, feature inspection was presented as a means to check the functionality of a system, comparing the system features to the specified criteria. Furthermore the

general principles and considerations that are relevant for the elaboration of test data in the context of the evaluation of translators' aids were presented. It was shown that software engineering principles for test data elaboration, specifically identifying classes of valid and invalid system inputs, as well as specifying boundary cases and typical problematic test cases, can be applied for user-oriented evaluation. These principles help to increase the reliability of evaluation results. Characteristics of the three types of test data prominent in the language engineering area, that is, test corpora, test suites and test collections, were discussed. In order to prove the applicability of the test model for user-oriented testing of translators' aids' systems, practical testing was performed with Trados TWB4W and IBM TM/2. The experiences with testing showed, that in evaluation preceding purchase decisions, it is promising to start off with feature inspection and supplement the results with, above all, task-oriented and benchmark testing, while interface-driven testing only plays a minor role in evaluation preceding purchase decisions. The numerous experiences made with scenario testing pointed to the fact that it is problematic to arrive at reliable, valid and value relevant results. It was argued that for evaluation preceding purchase decisions, the scenario test has to be considered as means of validating results. Its role for evaluation preceding purchase decisions can, in its own right, be compared to the role of sensitivity analysis as it is performed in the decision analysis process. In other words, scenario tests can show whether the figures obtained during the various stages of evaluation really make sense in the specific practical working environment.

In chapter 5, it was demonstrated that a quantitative representation of software testing results is possible on the basis of a common decision theoretic utility model. Problems that have to be specifically considered during the assessment procedure in evaluating translators' aids systems were addressed and solutions proposed. A task-oriented assessment approach was developed that represents a combination between software quality models known from software engineering and problem-oriented analysis as it is known from decision theory. This involves the weighting of the different tasks under evaluation, and the development of separate quality trees for each task. Considering each task independently during assessment guarantees the independence between criteria, which is a precondition for applying the additive weighted model from decision analysis. The applicability of the assessment procedure was demonstrated on the basis of the results obtained through the tests performed with the two commercially available translators' aids' systems. One of the most important findings of the assessment example was that the task approach advocated and applied in this thesis was of fundamental importance, since it keeps quality trees manageable and increases the transparency of the assessment procedure. It would have been impossible to put the many evaluation relevant attributes into the same value tree. The more metrics

available the more difficult it proved to be to distribute weights in a way that mirrors the preferences of the decision maker. Check routines were developed to make sure that weighting truly reflects the decision maker's preferences. The assessment procedure allowed the comparison of the evaluation score of the two systems under evaluation in numerical terms, and thus proved to be invaluable whenever a translator or an organisation has to decide which of the systems on the market is the best choice for the specific context.

The success of applying this evaluation framework in future evaluation contexts largely depends on the following factors:

(i) the comprehensive elicitation of domain truths
(ii) the coverage of metrics representative for the domain
(iii) the reliability and validity of test results
(iv) the responsible weighting process representing the preferences of the client of the evaluation process

Future evaluation of translators' aids' systems will benefit from the parameters for the elicitation of domain truths in the translation domain. One of the major achievements of this framework is to describe ways of how to arrive at measurable primitives from user requirements by means of modelling. Also, the goal-oriented model of test types provides guidance for future evaluation work. Last but not least, considering that, so far, evaluation models for translators' aids' systems have rested mainly on qualitative notions, another major achievement is that the present framework allows the quantitative assessment of evaluation results, also integrating the cost factor into the assessment procedure.

Future application of the evaluation framework in different situations will inevitably lead to a refinement of the procedures described in this thesis. While the application area for the evaluation framework in this thesis was that of translators' aids' systems it is envisaged that the framework can also be applied to the evaluation of other complex interactive systems.

As the practical testing and assessment procedures showed, evaluation performed along the framework developed above is possible but time-consuming. It is, at the moment, geared rather to the large translation industry market or to evaluation in international projects. In future, special attention must be paid to the question, how the effort of evaluation can be reduced without disturbing the reliability and validity of results.

The elaboration of typical user profiles along the featurisation model, depicting the typical tasks and related metrics could in future largely reduce the evaluation effort. Furthermore, re-usability could in future also be achieved on the level of metrics, test data or test programs. Elaboration, collection and presentation of these types of data for specific application areas in WWW sites would reduce evaluation effort to a large extent.

On the basis of the detailed description of the elements of the evaluation process and their relationships advocated in this thesis, the formalisation of the evaluation process is also conceivable in form of feature structures. Future research in this area may even lead to the automation of a large extent of the evaluation process.

To conclude, digging into three disciplines at a time is a complex matter and always runs the risk of yielding reproaches concerning a lack of depth in each discipline. The strong practical interest which has driven the theoretical development of the framework, however, should justify an interdisciplinary approach, notwithstanding the shortcomings such an endeavour might impose.

APPENDICES

Appendix 1: Excerpt of Result Report TWB Project

Appendix 1: TWB RESULT REPORT TEST CYCLE 1									
TOOL	Keyterm		Organisation	Debis		EVALUATORS		MB, SITE, SdT	
FUNCTION	OBSERVATION	TEST TYPE	PROPOSALS for MODIFICATION	Priority	DEVELOPER REMARK	Dead-line	TEAM	MEASURED QUANTITY	LEVEL
Hits	It is not clear where the information displayed with the term in the “hits” window comes from and what it means	Scen	The user should be able to specify the information to be displayed in the “hits” window	3	Keylex will restrict display to essential data	1-4	MB SdT	desirable feature	I, F
Hits	No need to display number of hits when n=1	Sys	Shortest rout possible to target information	2	Will be considered	2-5	MB	Desirable feature	I, F
Term window	Simplify data entry for translators	Sys	Small window for term-to-term entry, plus possibility to expand information in successive windows (strategy: from minimum to maximum information)	1	To be implemented in Keylex	2-4	MB	Desirable feature	I, F
Term	Moving between windows is error prone and may cause data loss	Sys	Investigate ways to move between entry windows before saving	1	Problem known, stable version will follow	2-4	MB, SdT	Comfortable handling	I
Term	When no entry was found (sending a term from the WinWord document), capitalization of the German search term is lost	Sys	When the term appears in the Term window, capitalization should be kept, especially when creating a new record.	1	Accepted	2-4	MB	Failure	F

Appendix 2: Result Report for Evaluation Preceding Purchase Decisions

1. Result Reports Task 1 (t_1) : TM Preparation

System 1: Trados, TAlign; System 2: IBM, ITM (initial translation memory)

1.1 Suitability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
alignment program?	binary	1	100	1	100	insp	1	-
hardware requirements: 486? 8 MB RAM?	binary nominal 80 20	 1 x 80 1 x 20	100	 1 x 80 1 x 20	100	insp	2	-
Windows?	binary	0	0	1	100	insp	3	✓
text statistics calculating repetition rates?	binary	0	0	0	0	insp	4	-
punctuation recognition	binary	1	100	1	100	insp	5	-
mark-up recognition	binary	1	100	1	100	insp	6	-
recognition of special text elements: proper names codes numbers dates currencies tables figures	binary nominal 1 1 1 1 1 1 1	 1 1 1 1 1 1 1	7/7 = 100	 1 1 1 1 1 1 1	7/7 = 100	insp	7	-
alignment procedures: batch interactive	binary nominal 80 20	 1 x 80 1 x 20	100	 1 x 80 0	80	insp	8	✓
correction of segmentation possible?	binary	1	100	0	0	insp	9	✓
correction of alignment possible?	binary	1	100	1	100	insp	10	-
alignment rate: number of aligned segments / number of segments setup 1 setup 2 total	ratio (%) 80 20	 92.30 81.81	 73.84 + 16.36 = **90.20**	 94.87 72.72	 75.89 14.54 **90.43**	bench	11	✓
alignment success rate: number of correctly aligned / segments number of aligned segments setup 1 setup 2 total	ratio (%) 80 20	 91.66 96.29	 73.32 19.25 **92.57**	 91.89 62.49	 73.51 12.49 **86.00**	bench	12	✓
total success rate: number of correctly aligned segments / number of segments setup 1 setup 2 total	ratio % 80 20	 84.61 78.78	 67.68 15.75 **83.43**	 87.17 45.45	 69.73 9.09 **78.82**	bench	13	✓

1.2 Compliance

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
format of input text:	binary nominal		95		90	insp	14	✓
RTF	50	1 x 50		1 x 50				
SGML	20	1 x 20		1 x 20				
Interleaf	15	1 x 15		1 x 15				
WordPerfect	10	1 x 10		0				
AmiPro	5	0		1 x 5				
charactersets:	binary nominal		90		100	insp	15	✓
all European	70	1 x 70		1 x 70				
Greek	20	1 x 20		1 x 20				
Japanese	10	0		1 x 10				

1.3 Customisability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
user-definition of special text elements:	binary	1	100	0	0	insp	16	✓

1.4 Installability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
installation program	binary	0	0	1	100	task	17	✓
installation without knowledge of operating system possible?	binary	0	0	1	100	task	18	✓

2. Result Reports Task 2 (t2) : TM and Termbank Retrieval

System 1: Trados TWB4W, ß version; System 2: IBM TM/2 version 1.0

2.1 Suitability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
hardware requirements:	binary		3/3 =		3/3 =	insp	19	-
IBM compatible?	nominal	1	100	1	100			
486?		1		1				
8 MB RAM?		1		1				
operating systems	binary		1/2 =		1/2 =	insp	20	-
Windows	nominal	1	50	0	50			
OS/2		0		1				
networking	binary nominal		100		80	insp	21	✓
LAN	80	1 x 80		1 x 80				
Novell	20	1 x 20		0				
importing aligned segments into database	binary	1	100	1	100	insp	22	-
accessing various databases during translation	binary	0	0	0	0	insp	23	-

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
translation retrieval from WinWord	binary	1	100	0	0	insp	24	✓
processing of special text elements: tables figures	binary nominal 1 0		1/2 = 50	 0 0	0/2 = 0	insp	25	✓
selection of source text segment: automatic manual	binary nominal 80 20	 1 x 80 1 x 20	100	 1 x 80 0	80	insp	26	✓
retrieval of parts of sentences	binary	1	100	0	0	insp	27	✓
retrieval of fuzzy matches	binary	1	100	1	100	insp	28	-
fuzzy match calculation	binary	1	100	0	0	insp	29	✓
setting of fuzzy match value	binary	1	100	0	0	insp	30	✓
display of fuzzy match percentage	binary	1	100	0	0	insp	31	✓
quality of fuzzy retrieval setup 1 setup 2 total	binary 50 50	 84.21 100	 42.10 50 **92.10**	 31.57 80.43	 15.78 40.21 **55.99**	bench	32	✓
constrained retrieval: fuzzy match setting only specific projects according to creation date	binary nominal	 1 1 1	3/3 = 100	0	0	insp	33	✓
automatic exchange of special text elements: numbers names dates tags time	binary nominal 50 10 20 10 10	 1 x 50 1 x 10 1 x 20 0 1 10	90	 1 x 50 1 x 10 0 1 x 10 0	70	task	34	✓
translation control information: creation user creation date change user change date usage counter project attributes source of info	binary nominal 1 1 1 1 1 1 0		6/7 = 85,7	 1 0 0 0 0 0 1	2/7 = 28,6	insp	35	✓
translation control information direct accessible during translation	binary	1	100	0	0	task interf	36	✓
system provides list of unfound terms?	binary	0	0	1	100	task interf	37	✓

2.2 Interoperability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
sharing of TM databases in network	binary	1	100	1	100	insp scen	38	-
retrieval of terms from termbases during translation memory retrieval	binary	1	100	1	100	insp	39	-
taking over of terminological info into translation text	binary	1	100	1	100	task	40	-

2.3 Compliance

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
translation retrieval editor: WinWord	binary	1	100	0	0	insp	41	✓
editing and formatting of translation proposals in text: WinWord functionality	binary	1	100	0	0	interf	42	✓
system messages standardised?	binary	1 (Windows)	100	0	0	interf	43	✓

2.4 Security

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
protection of TM databases within network	binary	1	100	0	0	insp interf	44	✓

2.5 Customisability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
order of translation proposals definable	binary	1	100	0	0	task	45	✓
user definition of control information (e.g. project codes, administration numbers)	binary	1	100	0	0	task	46	✓
retrieval of only exact matches possible?	binary	1	100	1	100	insp	47	-
user-definable special text elements?	binary	1	100	0	0	task	48	✓

2.6 Fault Tolerance

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
undo after inserting translations from memory	binary	1	100	0	0	task interf scen	49	✓
undo after inserting terminology from termbank	binary	1	100	0	0	task interf scen	50	✓
undo in translation editor?	binary	1	100	0	0	interf	51	✓

2.7 Understandability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
clarity of system layout	ordinal 1-5	5	100	2	40	task interf (scen) [1]	52	✓
understandability of interaction when using translation proposal	ordinal 1-5	5	100	3	60	task (scen)	53	✓
understandability of interaction when using terminology	ordinal 1-5	5	100	2	40	task (scen)	54	✓

2.8 Operability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
highlighted display of source text segment under consideration	binary	1	100	1	100	insp task	55	-
automatic initiation of translation retrieval after segmentation	binary	1	100	0	0	task	56	✓
input of special characters	binary	1	100	1	100	task	57	-
display of special characters in editor TM termbank	binary nominal	 1 1 1	3/3 = 100	 1 1 1	3/3 = 100	task	58	-
key combinations instead of direct manipulation possible?	binary	1	100	1	100	task scen	59	-
user-definable key combinations to functions	binary	0	0	1	100	task scen	60	✓
WYSIWYG in editor	binary	1	100	0	0	task scen	61	✓
access to detailed terminological information from editor possible	binary	1	100	1	100	task	62	-
taking over of terminology proposals easy?	ordinal 1-5	5	100	3	60	task	63	✓
taking over of translation proposals easy?	ordinal 1-5	5	100	2	40	task (scen)	64	✓

2.9 Time Behaviour

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
time (sec.) needed to retrieve translation and terms from memory	ratio v(5)=0	3	40	1	80	(scen) bench	65	✓

[1] note: the brackets around the scenario test type indicate that only qualitative results could be obtained, which, however, validate the values obtained by other test types.

2.10 Testability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
hotline support during testing available?	binary	1	100	1	100	task scen	66	-
change setups during testing straightforward?	ordinal 1-5	4	4/5 = 80	2	2/5 = 40	task interf	67	✓

2.11 Installability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
installation program	binary	1	100	1	100	task	68	-
time needed for installation	ratio v(60)=0	30 Min	50	50 Min	16,7	task	69	✓
installation without knowledge of operating system possible?	binary	1	100	1	100	task	70	-

3. Result Reports Task 3 (t_3) : Updating TM Databases

System 1: Trados TWB4W, ß version; System 2: IBM TM/2 version 1.0

3.1 Suitability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
modification of TM database after translation (correction etc.)	binary	1	100	0	0	task interf	71	✓
immediate updating procedure?	binary	1	100	1	100	task	72	-
option to keep back translation update	binary	0	0	0	0	interf	73	-
storage of translations in different databases possible?	binary	1	100	0	0	task	74	✓

3.2 Security

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
password check before modifying TM databases	binary	1	100	0	0	insp	75	✓

3.3 Fault Tolerance

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
undo after modifying translation memory	binary	1	100	0	0	task scen interf	76	✓

3.4 Understandability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
understandability of modification process	ordinal 1-5	4	4/5 = 80	0	0	task	77	✓

3.5 Learnability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
time needed to learn modification procedure	ratio v(30)=0	5 Min	83,3	0	0	task	78	✓

3.6 Operability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
modification process possible: while translating after finishing translation	binary nominal	 1 1	2/2 = 100	0	0	task	79	✓
steps needed to modify translation in TM database	ratio v(10)=0	5	50	0	0	task interf	80	✓

4. Result Reports Task 4 (t_4): Updating Termbanks

System 1: Trados TWB4W, ß version; System 2: IBM TM/2 version 1.0

4.1 Suitability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
modification of terms displayed for translation during translation process	binary	1	100	1	100	task interf	81	-
editing list of unfound terms?	binary	0	0	1	100	task	82	✓

4.2 Security

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
password check before modifying termbank	binary	1	100	0	0	task	83	✓

4.3 Fault Tolerance

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
confirmation after modifying termbank?	binary	1	100	1	100	interf	84	-

4.4 Understandability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
understandability of modification process	ordinal 1-5	4	4/5 = 80	3	60	task	85	✓

4.5 Learnability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
time needed to learn modification procedure	ratio v(30)=0	8 Min	73,4	10	66,7	task	86	✓

4.6 Operability

metric	scale	system 1 x	value v(x)	system 2 y	value v(y)	test type	No	rel
steps needed to modify terms while translating	ratio v(10)=0	5	50	5	50	task	87	-

Appendix 3: Test Data for Systematic Testing

1. Text Analysis as Preparation of Systematic Test for (t_1) and (t_2)

TEXT ANALYSIS PAIR 1 MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION AR27	OLD VERSION AR27
numbers in identical segments	1. GETRIEBE 722.6 2. GETRIEBE 722.620/621/622	1. GETRIEBE 722.3/4/5
identical parts of sentences	1. Ölstand nochmals PRÜFEN 2. Eine zu kleine BZW. zu große Ölmenge beeinträchtigt die Funktion des Getriebes 3. Bei kaltem Getriebe muß DIE ÖLSTANDSANZEIGE zwischen der "min." und "max." -Markierung, 25° (GETRIEBEÖLTEMPERATUR) liegen. 4. Bei betriebswarmen Getriebe muß DIE ÖLSTANDSANZEIGE an der "max"-Markierung, [80°] (GETRIEBEÖLTEMPERATUR) anliegen 5. Getriebeöl (AFT) nach Betriebsstoff-forschriften-Blatt NR. 236.10	1. Ölstand nochmals KONTROLLIEREN 2. Eine zu kleine, SO WIE EINE zu große Ölmenge beeinträchtigt die Funktion des Getriebes 3. Bei kaltem Getriebe (GETRIEBEÖLTEMPERATUR CA 30°) muß BEI RICHTIGEM ÖLSTAND DIE ANZEIGE zwischen der "min." und "max" -Markierung LIEGEN (BILD 3). 4. Bei betriebswarmen Getriebe (GETRIEBEÖLTERMPERATUR CA 80°) muß BEI RICHTIGEM ÖLSTAND DIE ANZEIGE an der "max"-Markierung anliegen (BILD 3). 5. AFT-ÖL nach Betriebsstoff-Vorschriften Blatt 236.4/6/7
left out/added segments	1. Das Fahrzeug muß waagrecht stehen 2. Getriebe auf Dichtheit prüfen 3. Bei Ölverlust Ursache ermittlen 4. Getriebeöl einfüllen 5. Ölmeßstab [(6)] bis zum Anschlag einstechen UND wieder herausziehen, Ölstand ablesen 6. Betriebsstoff-Vorschriften	1. Das Fahrzeug muß ZUR ÖLSTANDSKONTROLLE waagrecht stehen 2. Getriebe VOR ÖLSTANDSKONTROLLE auf Dichtheit prüfen 3. Bei größerem Ölverlust Ursache ermittlen 4. Getriebeöl BEI LAUFENDEM MOTOR einfüllen 5. Ölmeßstab (6) bis zum Anschlag einstechen [,] wieder herausziehen, Ölstand ablesen 6. AFT-ÖL NACH Betriebsstoff-Vorschriften BLATT [236.4/6/7]
identical individual terms	1. Getriebe 2. Handpumpe 3. Trichter	
identical segments	1. Sicherheitsvorschriften bei laufendem Motor beachten 2. Ggf. berichtigen 3. Zuviel eingefülltes Getriebeöl unbedingt ablassen oder absaugen,	

TEXT ANALYSIS PAIR 2: MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION ar2726	OLD VERSION ar2711
numbers	dates, construction numbers	
identical parts of sentences	1. DECKEL [(110)] ausklipsen (PFEILE)	1. ABDECKUNG [(3)] ausklipsen
left out/added segments	1. Abdeckung [(109)] ausklipsen (PFEILE) 2. Wählhebelgriff [(108)] ausbauen	1. Abdeckung [(3)] ausklipsen 2. Wählhebelgriff ausbauen
identical individual terms	1. Getriebe 2. Abdeckung 3. Schraube 4. Gabelkopf 5. Sicherung 6. Schaltwelle 7. Sperrhebel 8. Reparaturmittel 9. Nummer 10. Bezeichnung 11. Bestellnummer	
identical segments	1. Mittelschaltung zerlegen und zusammenbauen 2. zerlegen und zusammenbauen	

TEXT ANALYSIS PAIR 3 MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION AR2714	OLD VERSION AR2713
numbers	dates, construction numbers	
identical parts of sentences		
left out/added segments		
identical individual terms	1. Getriebe 2. Schaltstange 3. Stangenkopf	
identical segments	1. Schaltstange einstellen	

TEXT ANALYSIS PAIR 4: MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION AR2721	OLD VERSION AR276
numbers	dates, construction numbers	
different word order	1. Ölkühlerleitungen und Ölkühler spülen	1. Ölkühler MIT Ölkühlerleitungen spülen
identical parts of sentences		
left out/added segments		
identical individual terms	1: Handpumpe	
identical segments	1. Danach Ölkühler und Ölkühlerleitungen gründlich mit Druckluft ausblasen.	

TEXT ANALYSIS PAIR 5 MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION ar273	OLD VERSION ar274
numbers	dates, construction numbers	
combination of 2 into 1/splitting 1 into 2	1. Auspuffanlage [(94)] hinten mit einem Keilriemen abhängen	1.1 Auspuffanlage AN DER HINTEREN AUFHÄNGUNG AUSHÄNGEN. 1.2 MIT EINEM KEILRIEMEN ABHÄNGEN
different word order	1. Schaltstange (63) AUSBAUEN, DAZU Klips-sicherungen abnehmen	1. Klips-sicherungen[(32)] abnehmen UND Schaltstange AUSHÄNGEN
identical parts of sentences	1. Masseleitung AN Batterie abschließen 2. EINBAU in umgekehrter Reihenfolge 3. Ölstand im automatischen Getriebe prüfen, BZW ÖL EINFÜLLEN	1. Masseleitung DER Batterie abschließen 2. EINBAUEN in umgekehrter Reihenfolge 3. Ölstand im automatischen Getriebe prüfen, GGF.: RICHTIGSTELLEN
left out/added segments	1. Öleinfüllrohr (61) NUR VOM MOTOR abschrauben 2. Abschirmblech [(62)] ausbauen UND 13-POLIGE STECKUPPLUNG (26) TRENNEN 3. Sechskantschrauben [(95)] - Drehmomentwandler an Mitnehmerblech HERAUSSCHRAUBEN; DAZU ABDECKUNG (81) ausbauen 4. Ölablaßschraube [(9)] am Drehmomentwandler HERAUSDREHEN 5. Ölablaßschraube [(4)] an Ölwanne 6. Auspuffhalter [[64)] ausbauen 7. Drehmomentwandler herausnehmen	1. Öleinfüllrohr (15) abschrauben 2. Abschirmblech [(5)] ausbauen 3. Sechskantschrauben Drehmomentwandler an Mitnehmerblech 4. Ölablaßschraube am Drehmomentwandler 5. Ölablaßschraube an DER Ölwanne 6. Auspuffhalter [[10)] UND U-BÜGEL (11) ausbauen; DAZU SELBSTSICHERNDE MUTTERN (7) ABSCHRAUBEN 7. Drehmomentwandler herausnehmen, EINSETZEN
identical individual terms	1. Getriebe 2. Aus-, Einbauen 3. Nummer 4. Benennung 5. Haltegriff	
identical segments	1. Getriebe mit Drehmomentwandler aus-, einbauen. 2. Ölablaßschraube an der Ölwanne	

TEXT ANALYSIS PAIR 6 MB MANUAL		
TYPE OF SIMILARITY	NEW VERSION AR278	OLD VERSION AR275
numbers	dates, construction numbers	
only numbers different	1. Antriebsflansch [(96)] und Lagerzapfen-Kurbelwelle mit Molykote fetten.	1. Antriebsflansch [(32)] und Lagerzapfen-Kurbelwelle mit Molykote fetten.
combination of 2 into 1/splitting 1 into 2		
different word order	1. Drehmomentwandler beim Einsetzen hin-, und herdrehen, damit die Verzahnungen ineinandergreifen.	1. Beim Einsetzen Drehmomentwandler hin-, und herdrehen, damit die Verzahnungen ineinandergreifen
identical parts of sentences	1. ÖLABLAßSCHRAUBE AM Drehmomentwandler 2. BEI VERBRANNTEN ODER MIT ABRIEB DURCHSETZTEN GETRIEBEÖL, müssen Ölkühlerleitungen und Ölkühler gespült werden. 3. SIND IN DER GETRIEBEÖLWANNE Metallspäne, muß der Drehmomentwandler erneuert werden.	1. PRÜFWERTE Drehmomentwandler 2. RIECHT DAS GETRIEBEÖL VERBRANNT ODER IST ES MIT BELAGABRIEB DURCHSETZT, müssen DREHMOMENTWANDLER, Ölkühlerleitungen und Ölkühler gespült werden. 3. BEFINDEN SICH IM GETRIEBEÖL Metallspäne, muß der Drehmomentwandler erneuert werden.
left out/added segments	1. Molykote	1. Molykote -FETT
identical individual terms	1. Nummer 2. Benennung 3. Getriebe 4. Reparaturmittel 5. Bestell-Nummer 6. Haltegriff	
identical sentences	1. Drehmomentwandler herausnehmen, einsetzen 2. Herausnehmen 3. Getriebe senkrecht stellen 4. Drehmomentwandler herausziehen 5. Metallspäne werden durch Spülen nicht restlos beseitigt und können zu späteren Getriebeschäden führen. 6. Einsetzen	

2. Alignment Benchmarks Task 1

Setup 1 - German Source Text

AR27.00-0100A
Ölstand im automatischen Getriebe prüfen, ggf. richtigstellen
1.12.93
GETRIEBE 722.3/4/5
P27.00-0201-01
p2700020101
P27.00-0202-01

p2700020201
Bild 1 links (bis 09/93)
6 Ölmeßstab
6a Verschlußhebel
Bild 2 rechts (ab 10/93)
6 Ölmeßstab
6a Verschlußhebel
6b Sicherungsstift
P27.00-0204-01
p2700020401
Bild 3

Prüfen

Fahrzeug zur Ölstandskontrolle waagrecht stellen
1
Getriebe vor der Ölstandskontrolle auf Dichtheit prüfen
Bei größerem Ölverlust Ursache ermitteln und beseitigen
Sicherheitsvorschriften bei laufendem Motor beachten!
AH00.00-1000-01Z
2
Motor laufenlassen
3.1
Verschlußhebel (6a) öffnen
bis 09/93 (Bild 1)
3.2
Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken, beide Teile entfernen und Verschlußhebel (6a) öffnen
ab 10/93 (Bild 2)
4
Ölmeßstab (6) herausziehen
Mit fusselfreiem Tuch abwischen
5
Ölmeßstab (6) bis zum Anschlag einstecken, wieder herausziehen, Ölstand ablesen
Bei kaltem Getriebe (Getriebeöltemperatur ca. 30°C) muß bei richtigem Ölstand die Anzeige zwischen der "min." und "max."-Markierung liegen (Bild 3).
Bei betriebswarmen Getriebe (Getriebeöltemperatur ca 80°C) muß bei richtigem Ölstand die Anzeige an der "max."-Markierung anliegen (Bild 3)

Richtigstellen

7
Getriebeöl bei laufendem Motor einfüllen

126 589 12 63 00
Zuviel eingefülltes Getriebeöl unbedingt ablassen oder absaugen,
Eine zu kleine, so wie eine zu große Ölmenge beeinträchtigt die Funktion des Getriebes.
112 589 00 72 00
ATF-Öl nach Betriebsstoff-Vorschriften Blatt 236.4/6/7
8
Ölstand nochmals kontrollieren
Ggf. berichtigen
9.1
Verschlußhebel (6a) schließen
bis 09/93 (Bild 1)
9.2
Verschlußhebel (6a) schließen und Sicherungsstift (6b) einsetzen, bis er einrastet
ab 10/93 (Bild 2)
Ölmeßstab (6) so einsetzen, daß der Verschlußbügel (6a) gut zugänglich ist und nirgends ansteht.
126 589 12 63 00
126589126300
112 589 00 72 00
112589007200
Handpumpe
Trichter

Setup 1 – English Translation

AR27.00-0100A
Checking oil level in automatic transmission and correcting, if necessary
1.12.93
TRANSMISSION 722.3/4/5
P27.00-0201-01
p2700020101
P27.00-0202-01
p2700020201
Left-hand illustration 1 (up to 09/93)
6 Oil dipstick
6a Locking lever
Right-hand illustration 2 (as of 10/93)
6 Oil dipstick
6a Locking lever

6b Locking pin

P27.00-0204-01

p2700020401

Illustration 3

Checking

Park vehicle on a level surface to check the oil level

1

Check transmission for leaks before checking the oil level.

If oil loss is severe, determine and eliminate cause

Comply with safety regulations for running engine

AH00.00-1000-01Z

2

Allow engine to run

3.1

Open locking lever (6a)

Up to 09/93 (illustration 1)

3.2

Press locking pin (6b) to one side in direction of the arrow, remove both parts and open locking lever (6a)

As of 10/93 (illustration 2)

4

Pull out oil dipstick (6)

Wipe with fluff-free cloth

5

Insert oil dipstick (6) up to the stop, pull out again and read off oil level

When the transmission is cold (transmission oil temperature approx. 30°C) the display must be between the "min." and "max." marks for the correct oil level (illustration 3).

When the transmission is at operating temperature (transmission oil temperature approx. 80°C) the display must be at the "max." mark for the correct oil level (illustration 3).

Correcting

7

Pour in transmission oil when engine is running

126 589 12 63 00

Excess transmission oil must be drained or extracted

An insufficient or excessive quantity of oil impairs the operation of the transmission.

112 589 00 72 00

Automatic transmission fluid in accordance with Specifications for Service Products, sheet 236.4/6/7

8
Check oil level once again
Correct, if necessary
9.1
Close locking lever (6a)
Up to 09/93 (illustration 1)
9.2
Close locking lever (6a) and insert locking pin (6b) until it engages
As of 10/93 (illustration 2)
Insert oil dipstick (6) so that the locking clip (6a) is easily accessible and does not make contact anywhere.
126 589 12 63 00
126589126300
112 589 00 72 00
112589007200
Hand pump
Funnel

Setup 2 – English Source Text: Note: the layout of the text was taken over from the original!

2639/93EN
Answer given by Mr Flynn
on behalf of the Commission

1. Yes.

2. The Commission's policy on packaging is reflected in the philosophy and contents of its amended proposal on packaging and packaging waste[1]. This proposal is currently under discussion in the Council. This policy provides for measures for the prevention of the production of packaging waste and for the promotion of return, reuse and recovery operations.

   The proposal is based on the principle of conditional equivalence between packaging systems (reusable - one-way) as long as all comply with the established requirements, and under the condition that a return system has been set up for the effective recovery, and in particular recycling, of one-way packaging, and as long as life-cycle assessments justify no clear hierarchy. At this stage it is not possible to establish a general preference based on the ecological qualities of these different packaging systems.

   Accordingly, reuse systems are considered as a valid part of a packaging and packaging waste policy but it is not possible at this stage to claim a general environmental advantage for reuse over one-way systems which might be used as a valid argument in the particular case presented in the question.

[1] COM(93)418 final.

It is up to local authorities to evaluate, in the particular local conditions and in relation to the alternative solution, the possible effects on the environment, which should be considered as an element in making the decision.

3. Unemployment has reached unacceptable levels throughout the Community. For this reason, the Commission adopted in May 1993 a communication on a Community-Wide Framework for Employment[1], setting out a common framework for policy action in favour of employment creation. The purpose of this initiative is to put in place a strategic process for more concerted and collective action towards more employment-intensive growth. The aim is to focus on the employment problem, not just on the unemployment problem. This new focus aims to increase the overall employment intensity of production of goods and services, as well as to anticipate and accelerate new jobs and activities, address inequalities and raise the competitiveness of the Community's labour force. These aspects comprise an integral element of the White Paper on Growth, competitiveness and employment which the Commission presented to the Brussels European Council in December.

4. In accordance with the principle of subsidiarity, a decision on this point would fall under the responsibility of the local authorities. If a pilot scheme is adopted, it could apply, like other similar initiatives, for existing Community funds, in line with the particular rules of procedure for those funds.

./.

- 2 -

5. The principle of proximity, as established in the framework directive on waste, applies to the final disposal of waste and is therefore not relevant in this context.

6. The European Social Fund offers a number of options for ESF action to deal with the difficulties encountered, depending on the priorities put forward by the Dutch authorities:

– in the case of redundancies, Objective 3 allows ESF funding of training and retraining schemes in accordance with the new ESF regulations adopted by the Council on 20 July 1993. Article 1 of Regulation No 2084/93[2] refers to "persons exposed to long-term unemployment", i.e. those without work for more than 12 months or those unemployed for a lesser period but faced with a real danger of drifting into long-term unemployment. The Dutch Objective 3 plan for the coming period contains three major priorities of which one is the prevention and combatting of long-term employment and integration into the labour market of persons threatened with long-term unemployment. The plan includes training actions in this regard. Negotiations with the Dutch partners about the adoption of an Objective 3 Community Support Framework on the basis of the plan they have submitted will take place in the near future;

- training of workers to help them to adapt to industrial change will also be possible under the new Objective 4;

– part of the province of Friesland is eligible under Objective 5b for the coming 1994-1999 programming period. The Dutch government will forward in a very near future its proposals to the Commission. These proposals are expected to contain a request for ESF-support for employment growth and stability (in particular through continuing training and through guidance and counselling for workers of either sex, especially those in small and medium-sized enterprises and those threatened with unemployment, and for people who have lost their jobs).

More generally, it should be noted that the ERDF section of the Friesland operational programme could, as in the past, call for a range of measures to create new employment in the region. Specific measures for processing factories of agricultural products are covered by the horizontal measures under Objective 5a.

[1] COM(93)238 final
[2] OJ L 193, 31.07.1993.

Setup 2 – German Translation: Note: the layout of the text was taken over from the original!

2639/93DE
Antwort von Herrn Flynn
im Namen der Kommission
(6. Mai 1994)

Achtung: Bei den QE-Dokumenten darauf achten, daß Header (Dokumentnummer) und Footer bzw. Seitenzahl deaktiviert sind.
. Ja.

. Die Politik der Kommission im Bereich "Verpackungen" kommt zum Ausdruck in der Philosophie und im Inhalt ihres geänderten Vorschlags für eine Richtlinie über Verpackungen und Verpackungsabfälle [1]. Dieser Vorschlag wird zur Zeit beim Rat geprüft. Vorgesehen sind Maßnahmen zur Vermeidung von Verpackungsabfällen sowie zur Förderung der Rückgabe, Wiederverwendung und Verwertung von Abfällen.

Der Vorschlag beruht auf dem Grundsatz der Gleichwertigkeit der Verpackungssysteme (Wiederverwendung - Einwegsystem), sofern alle Systeme den gestellten Anforderungen gerecht werden, ein Rückgabesystem geschaffen worden ist, das eine tatsächliche Verwertung, insbesondere das Recycling, von Einwegverpackungen ermöglicht, und solange Lebenszyklusuntersuchungen keine klare Rangfolge erkennen lassen. Zur Zeit ist es noch nicht möglich, generell zu bestimmen, welches Verpackungssystem aufgrund seiner umweltschonenden Eigenschaften vorzuziehen ist.

Wiederverwendungssysteme bilden daher einen wertvollen Aspekt der Politik im Bereich "Verpackung und Verpackungsabfälle". Zur Zeit weiß man jedoch noch nicht, ob wiederverwendbare Verpackungen Einwegsystemen gegenüber generell von Vorteil für die Umwelt sind; im vorliegenden Fall kann dieses Argument daher nicht geltend gemacht werden.

Es obliegt den Lokalbehörden, unter Berücksichtigung der örtlichen Verhältnisse und der Alternativlösung mögliche Auswirkungen auf die Umwelt zu beurteilen, die bei der Beschlußfassung mitberücksichtigt werden sollten.

. Die Arbeitslosigkeit hat in der gesamten Gemeinschaft ein unannehmbares Maß erreicht. Aus diesem Grunde nahm die Kommission im Mai 1993 eine Mitteilung über einen gemeinschaftsweiten Rahmen für die Beschäftigung[2] an, die einen gemeinsamen Rahmen für politische Aktionen zur Förderung der Schaffung von Arbeitsplätzen festsetzt. Bezweckt wird die Einleitung eines strategischen Prozesses besser aufeinander abgestimmter gemeinsamer

1 KOM(93) 416 endg.

2 KOM(93) 238 endg.

Maßnahmen zur Förderung eines beschäftigungsintensiven Wachstums. Neben dem Problem der Arbeitslosigkeit soll ebenfalls das Beschäftigungsproblem angegangen werden. Dank dieses neuen Ansatzes sollen die Beschäftigungsmöglichkeiten bei der Produktion von Waren und der Erbringung von Dienstleistungen global verbessert werden, soll vorausschauend die beschleunigte Schaffung neuer Arbeitsplätze und Arbeitsbereiche bewirkt werden, und sollen Ungleichheiten angegangen sowie die Wettbewerbsfähigkeit der gemeinschaftlichen Arbeitskräfte verbessert werden. Diese Aspekte sind Bestandteil des Weißbuches über Wachstum, Wettbewerbsfähigkeit und Beschäftigung, das die Kommission dem Europäischen Rat im Dezember vorgelegt hat.

. Aufgrund des Subsidiaritätsprinzips sind die Gebietskörperschaften für eine solche Entscheidung zuständig. Sollte ein Pilotvorhaben beschlossen werden, so könnten, wie bei vergleichbaren Vorhaben, Gemeinschaftsmittel gemäß den einschlägigen Verfahren beantragt werden.

. Das in der Rahmenrichtlinie über Abfälle enthaltene Näheprinzip betrifft die endgültige Entsorgung und ist daher hier irrelevant.

. Der Europäische Sozialfonds (ESF) bietet je nach Art der von den niederländischen Behörden festgelegten Prioritäten eine Reihe von Interventionsmöglichkeiten:

- Bei Arbeitslosigkeit ermöglicht Ziel 3 die Bereitstellung von ESF-Mitteln zur Finanzierung von Ausbildungs- und Umschulungsmaßnahmen gemäß der neuen ESF-Verordnung, die am 20. Juli 1993 vom Rat erlassen wurde. In Artikel 1 der Verordnung Nr. 2084/93[1] ist die Rede von "Personen, die der Langzeitarbeitslosigkeit ausgesetzt sind", d.h. von Personen, die seit über 12 Monaten arbeitslos sind, oder die noch nicht so lange arbeitslos sind, jedoch Gefahr laufen, langfristig arbeitslos zu bleiben. Der niederländische Plan nach Ziel 3 für den kommenden Zeitraum umfaßt drei wesentliche Prioritäten, von denen eine die Vermeidung und Bekämpfung von Langzeitarbeitslosigkeit sowie die berufliche Eingliederung von Personen betrifft, die Gefahr laufen, langfristig arbeitslos zu bleiben. Der Plan umfaßt entsprechende Ausbildungsmaßnahmen. Die Verhandlungen mit den niederländischen Partnern im Hinblick auf die Annahme eines gemeinschaftlichen Förderkonzeptes nach Ziel 3 auf der Grundlage des von ihnen unterbreiteten Plans werden demnächst beginnen.

- Die Ausbildung von Arbeitskräften, um ihnen dabei zu helfen, sich auf den industriellen Wandel einzustellen, wird ebenfalls nach dem neuen Ziel 4 möglich sein.

- Für die Programmplanung 1994-1995 ist ein Teil der Provinz Friesland nach Ziel 5b förderungswürdig. Die niederländische Regierung wird ihre Vorschläge demnächst bei der Kommission einreichen. Diese Vorschläge werden voraussichtlich einen Antrag auf ESF-Förderung von Beschäftigungswachstum und -stabilität enthalten (vor allem durch Weiterbildung, Orientierung und Beratung der Arbeiskräfte jeglichen Geschlechts, insbesondere in kleinen und mittleren Unternehmen, der von Arbeitslosigkeit bedrohten Arbeitskräfte wie auch derjenigen, die ihren Arbeitsplatz verloren haben).

Generell ist zu bemerken, daß der EFRE-Teil des friesischen operationellen Programms, wie bereits früher auch, eine Reihe von Maßnahmen zur Schaffung neuer Beschäftigungsmöglichkeiten

[1] ABl. Nr. L 193 vom 31.7.1993

in der Region enthalten. Besondere Maßnahmen für Betriebe, die landwirtschaftliche Erzeugnisse verarbeiten, sind durch die horizontalen Maßnahmen nach Ziel 5a abgedeckt.

2. Retrieval Benchmark Benchmarks Task 2

Setup 1 – New Version of Aligned German Car Manual

AR27.00-0101A
Ölstand im automatischen Getriebe prüfen, bzw. Öl einfüllen
24.1.95
GETRIEBE 722.6
P27.50-0274-06
p2750027406
A 25° C (77° F)
B 80° C (180° F)
Das Fahrzeug muß waagrecht stehen
Getriebeöl einfüllen
1
Sicherungsstift (93a) entfernen, dazu die Platte des Sicherungsstiftes mit geeigneten Werkzeug abbrechen und den in der Verschlußkappe verbleibenden Stift nach unten herausdrücken
2
Verschlußkappe (93) abnehmen
3
Getriebeöl einfüllen
Getriebeöl (ATF) nach Betriebsstoffvorschriften-Blatt Nr. 236.10
126 589 12 63 00
Bei Neubefüllung erst ca. 4l Getriebeöl einfüllen
BF27.00-1001-01C
Sicherheitsvorschriften bei laufendem Motor beachten
AH00.00-1000-01Z
4
Motor starten und in Wählhebelstellung "P" bei Leerlaufdrehzahl laufen lassen
Bei Neubefüllung Rest der vorgeschriebenen Ölmenge nachfüllen
5
Fahrstufen bei stehendem Fahrzeug und Leerlaufdrehzahl des Motors mehrmahls durchschalten
Dabei Betriebsbremse betätigen
Ölstand prüfen, ggf. richtigstellen

Mit dem HHT kann die aktuelle Getriebeöltemperatur in den Wählhebelstellungen R, D, 4, 3, 2 und 1 ausgelesen werden
Dabei Betriebsbremse betätigen
6
Ölmeßstab bis zum Anschlag einstecken und wieder herausziehen, Ölstand ablesen
Bei kaltem Getriebe muß die Ölstandsanzeige zwischen der "min." und "max."-Markierung , 25° C (Getriebeöltemperatur) liegen.
Bei betriebswarmen Getriebe muß die Ölstandsanzeige an der "max."-Markierung, 80°C (Getriebeöltemperatur) anliegen.
140 589 15 21 00
Zuviel eingefülltes Getriebeöl unbedingt ablassen oder absaugen. Eine zu kleine, bzw. zu große Ölmenge beeinträchtigt die Funktion des Getriebes.
210 589 00 71 00
7
Ölstand nochmals prüfen
Ggf. berichtigen
8
Verschlußkappe (93) auf Öleinfüllrohr aufsetzen und Sicherungsstift (93a) eindrücken, bis er einrastet
9
Getriebe auf Dichtheit prüfen
Bei Ölverlust Ursache ermitteln und beseitigen.
Füllmengen Automatisches Getriebe
Nummer
Benennung
Getriebe 722.620/621/622
BF27.00-1001-01C
Füllmenge
bei Neubefüllung
Liter
9,3
bei Ölwechsel
Liter
-
Betriebsstoff-Vorschriften
Blatt
236.10
210 589 00 71 00
210589007100

126 589 12 63 00
126589126300
140 589 15 21 00
140589152100
Handpumpe
Meßstab
Trichter

Setup 2 – Test Suite

Handling of only identical *parts* of sentences
Test Case 1: deletion of sub-clauses
1. Ölstand im automatischen Getriebe prüfen.
2. Bei größerem Ölverlust Ursache ermitteln.
3. Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken.
4. Ölmeßstab (6) bis zum Anschlag einstecken.
5. Sicherungsstift (6b) einsetzen, bis er einrastet.

Handling of variations in sentence structure
Test Case 2: split sentence with two segments into two separate sentences
1. Ölstand im automatischen Getriebe prüfen. Ölstand im automatischen Getriebe richtigstellen.
2. Bei größerem Ölverlust Ursache ermitteln. Bei größerem Ölverlust Ursache beseitigen.
3. Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken. Beide Teile entfernen und Verschlußhebel (6a) öffnen.
4. Ölmeßstab (6) bis zum Anschlag einstecken. Ölmeßstab (6) herausziehen, Ölstand ablesen.
5. Verschlußhebel (6a) schließen. Sicherungsstift (6b) einsetzen, bis er einrastet.

Test Case 3: unite two separate sentences into one sentence
1. Fahrzeug zur Ölstandskontrolle waagrecht stellen (1) und Getriebe vor der Ölstandskontrolle auf Dichtheit prüfen.
2. Motor laufenlassen (3.1) und Verschlußhebel (6a) öffnen.
3. Ölmeßstab (6) herausziehen und mit fusselfreiem Tuch abwischen.
4. Bei kaltem Getriebe (Getriebeöltemperatur ca. 30°C) muß bei richtigem Ölstand die Anzeige zwischen der "min." und "max."-Markierung liegen (Bild 3) und bei betriebswarmen Getriebe (Getriebeöltemperatur ca 80°C) muß bei richtigem Ölstand die Anzeige an der "max."-Markierung anliegen (Bild 3).
5. Ölstand nochmals kontrollieren und ggf. berichtigen.

Test Case 4: change of sequence of main and sub-clauses
1. Ölstand im automatischen Getriebe richtigstellen, ggf. prüfen.
2. Bei größerem Ölverlust Ursache beseitigen und ermitteln.
3. Beide Teile entfernen und Verschlußhebel (6a) öffnen und Sicherungsstift (6b) seitlich in Pfeilrichtung wegdrücken.
4. Ölmeßstab (6) herausziehen und bis zum Anschlag einstecken.
5. Sicherungsstift (6b) einsetzen, bis er einrastet, Verschlußhebel (6a) schließen.

Handling of variable characters

Test Case 5: change names (no names in test text to be changed)

Test Case 6: change acronyms (only one available in test text)
1. ALP-Öl nach Betriebsstoff-Vorschriften Blatt 236.4/6/7

Handling of variable numbers

Test Case 7: change numbers (n+1)
1. *Bild 2 links (bis 09/93)*
2. Bild 3 rechts (ab 10/93)
3. Verschlußhebel (7a) öffnen
4. Ölmeßstab (7) bis zum Anschlag einstecken, wieder herausziehen, Ölstand ablesen
5. Bei kaltem Getriebe (Getriebeöltemperatur ca. 31°C) muß bei richtigem Ölstand die Anzeige zwischen der "min." und "max."-Markierung liegen (Bild 4).

Test Case 8: change date numbers (n+1)
1. *Bild 2 links (bis 10/94)*
2. Bild 3 rechts (ab 11/94)
3. bis 10/94 (Bild 1)
4. ab 11/94 (Bild 2)
5. bis 10/94 (Bild 1)

Handling of formatting

Test Case 9: for formatted text strings: remove formatting

1. GETRIEBE 722.3/4/5
2. Bild 1 links (bis 09/93)
3. Bild 3
4. Prüfen
5. Richtigstellen

Test Case 10: for formatted text strings: change formatting

1. GETRIEBE 722.3/4/5
2. **Bild 1 links (bis 09/93)**
3. **Bild 3**
4. *Prüfen*
5. *Richtigstellen*

Test Case 11: for non-formatted text strings add formatting

1. **Ölstand** im automatischen **Getriebe** prüfen, ggf. richtigstellen.
2. Bei größerem Ölverlust Ursache ermitteln und beseitigen.
3. Sicherungsstift (6b) seitlich in **Pfeilrichtung** wegdrücken, beide Teile entfernen und **Verschlußhebel** (6a) öffnen.
4. Ölmeßstab (6) bis zum **Anschlag** einstecken wieder herausziehen, **Ölstand** ablesen.
5. *Verschlußhebel* (6a) schließen und Sicherungsstift (6b) einsetzen, bis er einrastet.

Appendix 4: Test Data for Scenario Testing

CAT Course

Computer Aided Translation (CAT) Course
University of Helsinki
Department of Translation Studies
May 15-26, 1995
Lauri Carlson/Monika Höge

User Profile Form

Name:
Social Security Number:

Student profile:

Subjects	Years studied	Degree
Major subjects:		
Minor subjects:		
First		
Second		
Third		

Translator profile:

Languages	Years studied
First (native)	
Second	
Third	

Translation language pairs in order of preference

First
Second
Third

Special fields
First
Second
Third

Work experience as translator (duration, employer, language paris, special fields)

Experience with computers

Operating sytems

Basics of DOS (handling files and directories,starting and stopping programs ...)
Basics of Windows (handling mouse, icons, windows, menus, buttons)
Basics of Unix
Macintosh

Word processing

Program
WordPerfect
MS Word
AmiPro
Other (specify)

CAT Tools

Product Type	Product name	Used regularly	Tried out
Electronic word lists			
Electronic dictionaries (e.g. CD-ROM)			
Term banks			
Translation memories			
CAT programs			
Other (specify)			

Networking

User id in Helsinki University

Where	User id
Local net (Kouvola)	
PC net (Helsinki)	
Unix (Helsinki)	
Basics of email	
Baics of telnet	
Basics of ftp	
Baics of gopher (Heli)	
Basics of www	

Training Text: part of manual of TM/2 for Windows

To create a new terms list, select: Create list of new terms creating a list of new terms
Select this option to create a list of new terms.

The system provides the folder name as the default name for the list to be generated. You can specify any other name. For this type of list, you can also select Include context information. If a new term is found, the system can save the original segment (containing the term) as context information. This option is useful if you intend to copy the new terms to a dictionary that can contain context information.

Minimum number of occurrences:
Specify how often a term must occur in the document so that it is included in the list. To create the list of all terms of a document that are also in selected dictionaries, select: Create list of found terms creating a list of found terms. Select this option to create a list of found terms. The system provides the folder name as the default name for the list to be generated. You can specify any other name. For this type of list, you can also select: Include context information. If a term is found in one of the selected dictionaries, the system can save the original segment (containing the term)as context information. This option is useful if you intend to copy the new terms to a dictionary that can contain context information.

Add found terms to dictionary: To copy the found terms to a dictionary, select a dictionary from this list box or type the name of an existing dictionary.

For both new terms and found terms lists, you must specify: Dictionaries to be used for analysis. Select the dictionaries to be used for generating terminology lists from the list in the order in which you select them, and the dictionaries are searched in this order. You can select up to 10 dictionaries.

If needed, you can limit the generation of terminology lists by the following options:
Use exclusion lists: If you have terms you want to exclude from the lists to be generated, these terms must be put into an exclusion list. For each language for which you installed the language support, tm4w already provides an exclusion list. It contains so-called noise terms. Select the exclusion lists to be used from the list box. Use exclusion dictionaries. If you have a dictionary containing well-defined terms that you want to exclude from the terminology lists to be generated, select it from the list box. Click on Set to return to the Analyze Documents window.

To begin analysis, click on Analyze.

The document is segmented.
Depending on the options you selected, new terms lists and found terms lists are created, and can be modified and used for dictionary updates.

Text used in Scenario Test

```
{h2 id=dicnew}Creating a new dictionary in [tm4w]
/*---------------------------------------------------------------------
{i2 refid=new}a new dictionary
{i2 refid=dic}setting up a new one
/* {psc proc=host}
{p}
[tm4w] offers you several ways of setting up a new dictionary.
{ul}
{li}If you do not have any existing terminology
in machine-readable form,
you must create a completely new dictionary.
You do this by determining the dictionary properties.
In particular,
you must define a dictionary structure.
You can use a default structure offered by [tm4w],
or you can use the structure of an existing dictionary
in [tm4w] and change it.
A newly created dictionary is empty at first but you
can add entries
from a new terms list built during document analysis
or at any stage during the translation process.
In this way you can create dictionaries
that contain only terms oriented towards
specific translation tasks.
{li}
During analysis, [tm4w] can generate a found terms list
that contains all terms of the document that exist
in the referenced dictionaries.
[tm4w] can also copy the entry data of these
terms into a separate dictionary.
{li}
If you have your own terminology in a format of your own,
you must generate an external SGML-based dictionary
and you must import it into [tm4w].
In this case, a new dictionary is created with your terminology
and the entry structure as defined in the SGML file is taken.
{eul}
{p}
If you create a new dictionary via the {hp2}New Dictionary{ehp2}
window in [tm4w]
and you do not use the modelling option,
the following entry fields are offered as default fields[colon]
{p}
```

```
{table width=column cols='3*  *' scale='.8'
 concat=yes split=yes hp='0 0 0'}
{thd}{c}Entry field                {c}Level
{ethd}
{i2 refid=dic}default entry fields
{row}{c}Headword   *)              {c}entry
{row}{c}Part of Speech             {c}homonym
{row}{c}Abbrev./Fullform  *)       {c}sense
{row}{c}Definition                 {c}sense
{row}{c}Synonym  *)                {c}sense
{row}{c}Other Related Terms *)     {c}sense
{row}{c}Context                    {c}sense
{row}{c}Translation                {c}target
{row}{c}Company/Subject Code       {c}target
{etable}
{p
The entry fields marked with :xph}*){exph}
can be used as predefined search criteria
in the {hp2}Look up a Term{ehp2} window
(see {hdref refid=dicsrch}).
{p}
If you are working with a more comprehensive structure
and require more entry fields,
select {hp2}-Master-{ehp2} on the
{hp2}Use Existing Dictionary as Model{ehp2} window,
which offers an extensive dictionary structure.
You can rename or delete any fields
from this set of entry fields
and you can also add new user-defined fields to it.
/* {p}
/* In filters for lookup and printing you can use
/* all defined entry fields.
/cp
{p}
[tm4w] adds and updates time stamp information automatically,
provided the following date fields are selected
in the {hp2}New dictionary{ehp2} window
from the {hp1}-Master-{ehp1} model dictionary.
/*
{table width=column cols='* * 2*' scale='.8'
  concat=yes split=yes hp='0 0 0'}
{thd}
{c}Entry field
{c}Level
```

```
{c}Contents
{ethd}
{row}
{c}Creation Date
{c}entry
{c}The date when a headword was added to a dictionary
{row}
{c}Last Update
{c}sense
{c}The date when information at the sense level
of an entry was added or modified
{row}
{c}Creation Date
{c}target
{c}The date when a translation for a headword was added
{row}
{c}Last Update
{c}target
{c}The date when a translation entry field was last updated
{etable}
/* {p}
/* When you create a new dictionary,
/* you can protect it with a password against unauthorized changes.
{p}
{grid refid=gr01}
/*---------------------------------------------------- Prerequisites --
{gridseg}
{gridarea}
{p}{hp3}Prerequisites{ehp3}
{gridarea}
{p}None.
/*---------------------------------------------------- Calling sequence
{gridseg}
{gridarea}
{p}{hp3}Calling sequence{ehp3}
{gridarea shade=xlight}
{p}Select[colon]
{ol compact}
{li}The {hp2}Dictionary List{ehp2} window
{li}{hp2}New[ellip]{ehp2} from the {hp2}File{ehp2} menu
{eol}
{gridseg}
{gridarea}
{gridarea}
```

```
{p}The {hp2}New Dictionary{ehp2} window is displayed
(see {figref refid=dicnew1 page=no}).
/*-------------------------------------------------- Window ---------
{fig id=dicnew1 width=column  place=inline}
{i2 refid=win}New Dictionary
{figcap}New Dictionary window
{artwork name=eqfb7s4b width=75mm}
/* {screen}
/*  New Dictionary
/*  [separ]
/*  [vellip]
/*  [vellip]
/* {escreen}
{efig}
/*-------------------------------------------------- Options/parameters
{gridseg}
{gridarea}
{p}{hp3}Options and parameters{ehp3}
{gridarea}
/*
{parml}
{pt}Name
{pd}Enter a name of your choice for the new dictionary.
This name can be up to 8 alphanumeric characters long.
/*
{pt}Description
{pd}Type a description for the new dictionary.
The description can be up to 40 alphanumeric characters long.
/*
{pt}Location of dictionary
{pd}Specify where to place the new dictionary.
{p}
Select the drive on which you want the new dictionary to reside.
A dictionary grows with time, so select a drive with
enough space.
/*
{pt}Source Language
{pd}Select a source language from the list of installed languages
displayed in the list box.
/*
{pt}Use existing dictionary as model
{pd}If you do not want to determine
the dictionary entry structure yourself,
you can use the structure of an existing dictionary as a model by
```

clicking on {hp2}Yes[ellip]{ehp2}.
This takes you to the
{hp2}Use Existing Dictionary as Model{ehp2} window
where you can select a dictionary as model.
Click on {hp2}Select{ehp2} or {hp2}Cancel{ehp2} to return
to the {hp2}New Dictionary{ehp2} window.
For more information on this option see {hdref refid=dicnew4}.
{pt}Change entry fields
{pd}If you want to change the dictionary entry structure
(add, delete, or rename entry fields),
click on {hp2}Yes[ellip]{ehp2}

Evaluation Description Sheet Scenario Test

EVALUATION DESCRIPTION Scenario Test					
organisation	Uni Helsinki	date	26/05 1995	test ID	1.1
MOTIVATION					
perspective	task-oriented				
interest	scientific and practical comparison of TM/2 with Trados TWB				
consumer	academia and users				
SYSTEMS					
name	TWB4W	version	ß		
name	TM for Windows	version	1.0		
hardware platform	LAN with 10 PCs 386, 8 RAM				
software modules	translation memories, termbanks, editor				
ENVIRONMENT					
test personnel evaluators	M. Höge, L. Carlson observers: 10 Finnish Students				
subjects	10 Finnish Students of Translation with English as Major or Minor				
budget	1 PM				
time	04 - 06 1995				
QUALITY					
characteristics	suitability, fault tolerance, understandability, learnability, operability, time behaviour,				
view on quality	black box				
type of metrics	qualitative				
TYPE OF EVALUATION	comparative adequacy evaluation				
TESTING					
test type:	scenario testing				
instruments	user profile questionnaire, scenario checklist, observation, post-testing interview				
description	scenario test integrated into training environment				
data	test corpora TM/ manual				

References

Ackerman, A. F.; Fowler, P.J.; Ebenau, R.G.: "Software Inspection and the Industrial Production of Software" in: Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984, pp. 13 - 40.

Ahmad, K.; Holmes-Higgin, P.; Rogers, M.; Höge, M.; Le-Hong, K.; Huwig, C.; Kese, R.; Mayer, R.: "User-driven Software Development: Translator's Workbench - an exemplar case study" in: Smith, MJ. and Salvendy, G. (eds.): Advances in Human Factors/Ergonomics, 19A. Human-Computer Interaction: Applications and Case Studies. Proceedings of the fifth International Conference on Human-Computer Interaction, (HCI International '93), Orlando, Florida, August 8 - 13, 1993, Volume 1, pp. 319 - 324.

ALPAC: Languages and Machines: Computers in Translation and Linguistics. Report of the Automatic Language Processing Advisory Committee, Division of Behavioural Sciences, National Academy of Sciences, National Research Council Publication 1416, Washington, D.C., 1966.

ARPA: Proceedings of the Machine Translation Evaluation Workshop. Vienna, 1994.

Athappily, K.; Galbreath, R. S.: "Practical Methodology Simplifies DSS Software Evaluation Process" in: Data Management 24 (1986) 2, pp. 10-17.

Baird, J.C.; Noma, E.: Fundamentals of Scaling and Psychophysics. New York, 1978.

Balkan, L.; Meijer, S.; Arnold, D.; Dauphin, E.; Estival, D.; Falkedahl, K.; Lehmann, S.; Netter, K.; Regnier-Prost, S.: Issues in Test Suite Design. Report (D-WP2.1) to LRE 62-089 Test Suites for Natural Language Processing (TSNLP), University of Essex, 1994 - 1.

Balkan, L.; Meijer, S.; Arnold, D.; Dauphin, E.; Estival, D.; Falkedahl, K.; Lehmann, S.; Regnier-Prost, S.: Test Suite Design Guidelines and Methodology. Report (D-WP2.1) to LRE 62-089 Test Suites for Natural Language Processing (TSNLP), University of Essex, 1994 - 2.

Balkan, L. Meijer, S.; Arnold, D.; Dauphin, E.; Estival, D.; Falkedahl, K.; Lehmann, S.; Netter, K.; Oepen, S.; Regnier-Prost, S.: Test Suite Design Annotation

Scheme. Report (D-WP2.1) to LRE 62-089 Test Suites for Natural Language Processing (TSNLP), University of Essex, 1994 - 3.

Balzer, R.: Final Report on GIST. USC/ISI, Marina del Rey, Technical Report, 1981.

Basili, V.; Rombach, H.D.: "The TAME Project: Towards Improvement-Oriented Software Environments" in: IEEE Transactions on Software Engineering, Vol. 14, No. 6, June 1988, pp. 758 - 773.

Bechtel, W. (ed.): Integrating Scientific Disciplines. Science and Philosophy Series, Pittsburg, 1986.

Bechtel, W.: "The Nature of Scientific Integration" in: Bechtel, W. (ed.): Integrating Scientific Disciplines. Science and Philosophy Series, Pittsburg, 1986, p. 3 - 52.

Bell, T.E.; Bixler, D.C.; Dyer, M.E.: An Extendable Approach to Computer Aided Software Requirements Engineering" in: IEEE Transactions on Software Engineering, 1977, SE-3 (1), pp. 49 - 60.

Bell, J.; Hardimann, R.J.: "The Third Role - the Naturalistic Knowledge Engineer" in: Diaper, D. (ed.) Knowledge Elicitation: Principles, Techniques, Applications, 1989, pp. 49 - 85.

Berger, J.O.: Statistical Decision Theory. Foundations, Concepts, and Methods. New York, 1980.

Berkel, B. van; Smedt, K. de: "Triphone Analysis: A Combined Method for the Correction of Orthographical and Typographical Errors" in: Proceedings of the Second Conference on Applied Natural Language Processing, Austin, 9-12 February 1988 pp. 77-83.

Beylard-Ozeroff, A.; Králová J.; Moser-Mercher B. (eds.): Translator Strategies and Creativity. Selected Papers from the 9th International Conference on Translation and Interpreting. Prague, September 1995. In honor of Jirí Levý and Anton Popovic. Amsterdam 1998.

Bickerton, M.J.; Siddiqi, J.: "The Classification of Requirements Engineering Methods" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 182 - 186.

Birkenbihl, M.: Train the Trainer: Arbeitsbuch für Ausbilder und Dozenten mit 21 Rollenspielen und Fallstudien. Landberg/Lech, 1990.

Boehm, B.W.; Brown, J.R.; Lipow, M.: "Quantitative Evaluation of Software Quality" in: Proceedings of the 2nd International Conference on Software Engineering, 1976, pp. 592 - 605.

Boehm, B.W.; Brown, J.R.; Kaspar, H.; Lipow, M.; MacLeod, G.J.; Merrit, M.J.: Characteristics of Software Quality. TRW Series of Software Technology, Vol. 1, Amsterdam - New York - Oxford, 1978.

Boisen, S.; Bates, M.: "A Practical Methodology for the Evaluation of Spoken Language Systems." In: Proceedings of the Third Conference on Applied Natural Language Processing, Trento,1992, pp. 162-169.

Borgida, A.; Greenspan, S.; Mylopoulos, J.: "Knowldge Representation as a Basis for Requriements Specification" in: IEEE Computer 18 (4), pp. 82 - 91.

Bouyssou, D. (ed.): Preference Modelling. Bussum, 1998.

Breuker, J.A.; Weilinga, B.J.; van Someren, M.W.: The KADS System Functional Description. Esprit Project 1098, Deliverable T1.1, University of Amsterdam, 1986.

Brinkhoff, N.: "Towards Standards in Language Engineering: EAGLES" in: XIII Magazine, May 1993 Issue No.10, pp. 25 - 27.

Bubenko, J.A. jr: "Challenges in Requirements Engineering" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 160 - 162.

Bukowski, J.V.: "Evaluating Software Test Results: A New Approach" in: Proceedings Annual Reliability and Maintainability Symposium (1987), Philadelphia, USA, 27 -29. Jan. 1987, pp. 369-375.

Byrnes, J.P.: The Nature and Development of Decision Making. Mahwah, NJ, 1998.

Carroll, J.M.; Grudin, J.; McGrew, J.; Scapin, D.: "Task Analysis: the Oft Missing Step in the Development of Computer-Human Interfaces; Its Desirable Nature, Value, and Role" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 1051 - 1054

CETIL Systran Evaluation and Comparison. Summary Report of Rewisers' Comments on Machine Produced Translations. Working Document for the CETIL meeting 26 and 27 March 1979, CETIL/139/79, Luxembourg, 1979.

Chéhab, P.: "Vernetztes Denken - Praxis in der SWISSAIR Dargestellt am Beispiel der Überprüfund unserer Dienstleistung" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 181 - 193.

Chen, P.: "The Entity-Relationship Model - Toward a Unified View of Data" in: ACM Transactions on Database Systems 1 (1) March 1976, pp. 9 - 36.

Cheung, R.C.: "A User-Oriented Software Reliability Model" in: IEEE Transactions on Software Engineering,1980, pp. 118-125.

Chinchor, N.: "MUC-3 Evaluzations Metircs." In: Proceedings of the Third Message Understanding Conference (MUC-3), San Mateo, 1991, pp. 17-24.

Chomsky, N.: Aspects of the Theory of Syntax. Cambridge Mass., 1965.

Christ, M.L.; Itzfeld, W.D.; Schmidt, M.; Timm, M.; Watts, R.: Software Quality Measurement and Evaluation. Final Report Volume II Project MQ: Measuring Quality of Software Products and Software Production Aids. GMD, Sankt Augustin, FRG and NCC, Manchesterr, UK, 1984.

Chung, L.; Nixon, B.A.; Yu, E.: "Using Non-Functional Requirements to Systematically Support Change" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 132 - 139.

Coad, P.; Yourdon, E.: Object-Oriented Analysis. Englewood Cliffs, 1991².

Colgan, L.; Brouwer-Janse, M.: "An Analysis of the Circuit Design Process for a Complex Engineering Application" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 253 - 258.

Cook, V.H.; Hartrum, T.H.; Howatt, J.W.; Woffinden, D.S: "A Framework for Evaluating Software Development Methods" in: Proceedings of the IEEE 1988, National Aerospace and Electronics Conference: NAECON 1988, 23-27 May 1988, Dayton, OH, USA, 2 (1988), pp. 667-669.

Cook, V. (ed.).: Experimental Approaches to Second Language Learning. Oxford, 1986.

Cordingley, E.: "Knowledge Elicitation: Techniques for Knowledge Based Systems" in: Diaper, D. (ed.) Knowledge Elicitation: Principles, Techniques, Applications, 1989, pp. 89 - 172.

Crellin, J; Horn, T.; Preece, J.: "Evaluating Evaluation: A Case Study of the Use of Novel and Conventional Evaluation Techniques in a Small Company" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 329 - 335.

Dechert, H.W.; Sandrock, U.: "Thinking-aloud Protocols: the Decomposition of Language Processing." in: Cook, V. (ed.).: Experimental Approaches to Second Language Learning. Oxford, 1986.

Deming, W.E.: "The Logic of Evaluation" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 53 - 68.

Deutsch, M.S.: Software Verification and Validation. Englewood Cliffs, NJ 07632, 1982.

DGQ; NTG (eds.): Software Qualitätsicherung. Aufgaben, Möglichkeiten, Lösungen. Berlin/Offenback, 1986.

Diaper, D. (ed.): Knowledge Elicitation: Principles, Techniques, and Applications. Chichester, UK, 1989-1.

Diaper, D.: Task Analysis for Human Computer Interaction. Chapter 4. Ellis Horwood, 1989-2, p. 108 - 159.

Diaper, D.: Task Analysis for Human Computer Interaction. Chapter 7. Ellis Horwood, 1989-3, p. 210 - 251.

Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990.

Diaper, D.: "Analysing Focused Interview Data with Task Analysis for Knowledge Descriptions," in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 277 - 282.

Diller, H.-J.; Kornelius, J.: Linguistische Probleme der Übersetzung. Tübingen, 1978.

DIN 66234 Teil 8 in: DIN (1988): Bildschirmarbeitsplätze, Berlin.

Dörner, D.: Problemlösen als Informationsverarbeitung. Stuttgart, Berlin, Köln, Mainz, 1979/2.

Dollhoff, T.L.: "Evaluating Manufacturing Software" in: Production Engineering 32, (1985), Cleveland, pp. 68-72.

Douglas, S.: Requirements Analysis for Linguistic Engineering Evaluation. Edinburgh, 1995.

Downs, T.: "An Approach to the Modelling of Software Testing with some Applications" in: IEEE Transactions on Software Engineering,1985, pp. 375-386.

Dworatschek, S., Höcker, H.: "Möglichkeiten einer Bewertung softwaretechnologischer Methoden" in: Angewandte Informatik 27 (1985) 5, pp. 183-190.

Dzida, W.; Freitag, R.; Hoffmann, C.; Vlader, W.: "Bridging the Gap between Task Design and Interface Design." in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 239 - 245.

EAGLES MT Evaluation Working Group. EAGLES Evaluation of Natural Language Processing Systems. Final Report. EAGLES Document EAG-EWG-PR.2, ISBN 87-90708-00-8. Center for Sprogteknologi, Copenhagen, 1996.

Easterbrook, S.: "Domain Modelling with Hierarchies of Alternative Viewpoints" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 65 - 72.

Easterbrook, S.; Nuseibeh, B.: Managing Inconsistencies in an Evolving Specification" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 48 - 55.

Edwards, W.; Guttentag. M.; Snapper, K.: "A Decision-Theoretic Approach to Evaluation Research" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 139 - 181.

Edwards, W.; Newman, J.R.: "Multiattribute Evaluation" in: Sullivan, J.L.; Niemi, R.G.(eds.): Sage University Series on Quantitative Applications in the Social Sciences. Beverly Hills and London, 1982, p. 5 - 96.

El Emam, K.; Madhavji, N.H.: "A Field Study of Requirements Engineering Practices in Information Systems Development" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 68 - 80.

Ericsson, K.A.; Simon, H.A.: Protocol Analysis: Verbal Report as Data. London, 1985.

Fagan, M.E.: "Design and Code Inspection to Reduce Errors in Program Development" in: IBM System Journal, Vol 15, No.3, 1976.

Falkedal, K.: Evaluation Methods for Machine Translation Systems: An Historical Overview and a Critical Account. Report to Suisstera, ISSCO, Geneva, 1991.

Färch, C.; Kasper, G.: Introspection in Second Language Research. Clevedon, 1987.

Fickas, S.; Feather M.S.: "Requirements Monitoring in Dynamic Environments" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 140 - 147.

Flickinger, D.; Nerbonne, J.; Sag, I.; Wascow, T.: Toward Evaluation of NLP Systems. Hewlett Packard Laboratories, Palo Alto, CA, 1987.

Fluck, H.R.: Fachsprachen. Tübingen, 1985/3.

Freigang, K-H.: "Überlegungen zu einer theoretisch-linguistisch fundierten Methodologie der Übersetzungswissenschaft (1978)" in: Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt, 1981, pp. 150 - 168.

Freigang, K-H.; Reinke, U (eds.): Saarbrücker Studien zu Sprachverarbeitung und Übersetzen, Fachrichtung 8.6, angewandte Sprachwissenschaft sowie Übersetzen und Dolmetschen. Universität des Saarlandes1995.

French, S.: Decision Theory. An Introduction to the Mathematics of Rationality. London, 1986.

French, S.: Readings in Decision Analysis. London, 1989.

Fulford, H.; Höge, M.: Preliminary Study of User Requirements - Methods of Investigation. Internal Report of the ESPRIT II Project 2315 Translator's Workbench (TWB). Stuttgart and Guildford, 1989, unpublished.

Fulford, H.; Höge, M.; Ahmad, K.: User Requirements Study. Final Report of the ESPRIT II Project 2315 Translator's Workbench (TWB). Stuttgart and Guildford, 1990, unpublished.

Gaines, Brian R.: "A Methodological Framework for the Design and Evaluation of Software in Systems Involving Complex Human-Computer Interaction" in: Berichte des German Chapter of the ACM 29 (1987) Stuttgart, pp. 44-73.

Galliers, J.R.; Sparck Jones, K.: Evaluating Natural Language Processing Systems. Technical Report No. 291, University of Cambridge Computer Laboratory, 1993.

Gerloff, P.: "Identifying the Unit of Analysis in Translation: Some Uses of Think-aloud Protocols of Translation." in: Färch, C.; Kasper, G.: Introspection in Second Language Research. Clevedon, 1987.

Glickman, S.; Becker, M.: "A Methodology for Evaluating Software Tools" in: Conference on Software Tools (1985), New York, USA, April 15-17 1985, pp. 190-196.

Goguen, J.A.; Linde, C.: "Techniques for Requirements Elicitation" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 152 - 164.

Golub, A. L.: Decision Analysis: an Integrated Approach. New York, 1997.

Gomez, P.; Probst, G.J.B.: "Vernetztes Denken für die Strategische Führung eines Zeitschriftenverlags" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 23 - 39.

Gough, P.A.; Fodemski, F.T.; Higgins, S.A.; Ray, S.J.: "Scenarios, an Industrial Case Study and Hypermedia Enhancements" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 10 - 17.

Gotel, O.; Finkelstein, A.: "Contribution Structures" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 100 - 107.

Gourlay, J.S.: "Introduction to the Formal Treatment of Testing" in: Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984, pp. 67-73.

Greenspan, S.: "The Next 25 Years: New Customers, New Environments, New Requirements" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 36 - 37.

Gurel, L.: "The Human Side of Evaluating Human Services Programs: Problems and Prospects" in: Guttentag. M.; Struening, E. L. (eds.): Handbook of Evaluation Research. Vol. 2. Beverly Hills, London, 1975, p. 11 - 28.

Guttentag. M.; Struening, E. L. (eds.): Handbook of Evaluation Research. Vol. 2. Beverly Hills, London, 1975.

Harker, S.D.P.; Olphert, C.W.; Eason, K.D.: "The Development of Tools to Assist in Organisational Requirements Definition for Information Technology Systems," in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 295 - 300.

Harker, S.D.R.; Eason, K.D.; Dobson, J.E.: "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 266 - 272.

Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984.

Hausen, H.L.: "Comments on Practical Constraints of Software Validation Techniques" in: Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984, pp. 323 - 333

Hausen, H.L.; Müllerburg, M.: "Kombination von Verfahren für die Software-Prüfung" in: Internationaler Kongress für Datenverarbeitung und Informationstechnologie (IKD), 1982, pp. 111-125.

Hausen, H.L.; Müllerburg, M.; Schmidt, M.: "Über das Prüfen, Messen und Bewerten von Software. Methoden und Techniken der analytischen Software-Qualitätssicherung" in: Informatik Spektrum 10 (1987) 3, pp. 123-144.

Hayward, S.; Breuker, J.A.; Weilinga, B.J.: The KADS methodology: Analysis and design for knowledge bassed systems. ESPRIT P 1098, Deliverable Y1, STC Technology Ltd., 1987.

Heid, U.: Evaluation der französisch-deutschen SYSTRAN-übersetzung, Vorhabesskizze, IMS, Stuttgart, 1988.

Hewitt, J.; Hobson, J.; Sapsford-Francis, J.: "An Application of Task Analysis to the Development of a Generic Office Reference Model" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 265 - 269.

Höge, M.: Theory and Practice of Special Language Translation with a Focus on Terminology Support as the Principal Function of Computer Aided Translation Systems. M.A. Thesis, University of Stuttgart, 1989, unpublished.

Höge, M.; Wiedenmann, O.; Kroupa, E.: Evaluation of the TWB - Theoretical Framework and Practical Application. Report of the ESPRIT II Project 2315 Translator's Workbench (TWB), Mercedes-Benz AG Stuttgart, 1991, unpublished.

Höge, M.; Kroupa, E.: "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, (1991) pp. 1036 - 1040.

Höge, M.; Hohmann, A.; Mayer, R.: Evaluation of TWB - Operationalization and Test Results. Final Report of the ESPRIT II Project 2315 Translator's Workbench (TWB), Mercedes-Benz AG Stuttgart, 1992, unpublished.

Höge, M.; Hohmann, A.; van der Horst, K.; Evans, S.; Caeyers, H.: User Participation in the TWB II Project - the First Test Cycle. Report of the ESPRIT II Project 6005 Translator's Workbench II (TWB II), Mercedes-Benz AG Stuttgart, Paris, Luxembourg, 1993, unpublished.

Höge, M.; Hohmann, A.; Le-Hong, K.: "User-Centered Software Development and Evaluation" in: Poster Sessions. Abridged Proceedings of the fifth International Conference on Human-Computer Interaction, (HCI International '93), Orlando, Florida, August 8 - 13, 1993, p166.

Höge, M.; Hohmann, A.; Le-Hong, K.: "Key Players." in: Kugler, M.; Ahmad, K.; Thurmair, G.: (eds.): Translator's Workbench. Tools and Terminology for Translation and Text Processing. Research Reports ESPRIT Project 2315 TWB Volume 1, Brussels-Luxembourg, 1995, pp. 4-5.

Höge, M.; Ahmad, K.; Davies, A.; Fulford, H.; Holmes-Higgin, P.; Rogers, M.: "User Participation in Software Development." in: Kugler, M.; Ahmad, K.; Thurmair, G. (eds.): Translator's Workbench. Tools and Terminology for

Translation and Text Processing. Research Reports ESPRIT Project 2315 TWB Volume 1, Brussels-Luxembourg, 1995, pp. 8 - 15.

Höge, M.; Hohmann, A.; Le-Hong, K.: "Software Testing and User Reaction", in Kugler, M.; Ahmad, K.; Thurmair, G. (eds.): Translator's Workbench. Tools and Terminology for Translation and Text Processing. Research Reports ESPRIT Project 2315 TWB Volume 1, Brussels-Luxembourg, 1995, pp. 168 - 173.

Höge, M.:"A Framework for the Qualitative and Quantitative Evaluation of Translator's Aids Systems", in: Jacquemin, C.; Mariani, J.; Paroubek, P. (eds.): Proceedings of the LREC2000 satellite workshop, Using Evaluation whitin HLT Programs: Results and Trends, Athens, Greece, May 30th 2000, pp. 21-27.

Hogarth, R.M.: Judgement and Choice: The Psychology of Decision. Suffolk, 1985[3].

Hohnhold, I.: "Übersetzungsorientierte Terminologiearbeit" in: Lebende Sprachen 28 (1) pp. 2 - 8; (3) pp. 102 - 104; (4) pp. 145 - 148, 1983

Hohnhold, I.: Übersetzungsorientierte Terminologiearbeit. Eine Grundlegung für Praktiker. Stuttgart, 1990.

Hohmann, A.; Le-Hong, K.: Software Training for TWB - Theoretical Aspects in the Development of a Training Methodology. Report of the ESPRIT II Project 6005 Translator's Workbench II (TWB II). Stuttgart, 1993, unpublished.

Hohmann, A.; Le-Hong, K.: Software Training for TWB - Development of a Training Methodology and First Practical Results. Report of the ESPRIT II Project 6005 Translator's Workbench II (TWB II). Stuttgart, 1994, unpublished.

Hohmann, A.; Le-Hong, K.; van der Horst, K.: User Participation in the TWB II Project - the Second Test Cycle. Report of the ESPRIT II Project 6005 Translator's Workbench II (TWB II). Stuttgart, Luxembourg, 1994, unpublished.

Hölscher, A.; Möhle, D.: "Cognitive Plans in Translation." in: Färch, C.; Kasper, G.: Introspection in Second Language Research. Clevedon, 1987.

Hommel, G.; Krönig, D. (eds.): Requirements Engineering. Informatik-Fachberichte 74, Berlin-Heidelberg-New York, 1983.

Hönig, H.G.; Kußmaul, P.: Strategie der Übersetzung. Ein Lehr und Arbeitsbuch. Tübingen, 1982.

Hönig, H.G.: "Holmes' "Mapping Theory" and the Landscape of Mental Translation Processes" in: Van Leuven-Zwart, K.M.; Naaijkens, T. (eds.): Translation Studies: The State of the Art. Proceedings of the First James S Holmes Symposium on Translation Studies. Amsterdam, Atlanta, 1991, pp. 77 - 89.

Howden. W.E.: "Empirical Studies of Software Validation" in: Miller, E. and Howden, W.E.: Software Testing and Validation Techniques, IEEE, 1978.

Howden, W.E.: "Functional Program Testing" in: IEEE Transactions on Software Engineering SE-6, 1980, pp. 162 - 169.

Hughes, J.; O'Brien, J.; Rodden, T.; Rouncefield, M.; Sommerville, I.: Presenting Ethnography in the Requirements Process" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 27 - 34.

IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard 729-1983, New York, 1983.

IEEE Guide to Software Requirements Specifications. Std. 830-1984, New York, 1984.

IEEE Guide for Software Verification and Validation Plans. Std 1059-1993, New York, 1993.

IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1993.

Ip, W.K.; Damodaran, L.; Olphert, C.W.; Maguire, M.C.: "The Use of Task Allocation Charts in System Design: a Critical Appraisal," in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 289 - 294.

Isabelle, P.; Bourbeau, L.:"TAUM-AVIATION: its Technical Features and Some Experiemental Results" in: Slocum, J. (ed.): Machine Translation Systems. Cambridge, 1988, pp. 237 - 264.

ISO 8402 (Standard): Quality - Vocabulary. International Organisation for Standardization, 1986.

ISO 9000 (Standard): Quality Management and Quality Assurance Standards - Guidelines for Selection and Use. International Organisation for Standardization, 1987.

ISO 9000-3 (Standard): Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software. International Organisation for Standardization, 1991.

ISO 9001 (Standard): Quality Systems - Model for Quality Assurance in Design/Development, Production, Installation and Servicing. International Organisation for Standardization, 1988.

ISO 9004 (Standard): Quality Management and Quality System Elements - Guidelines. International Organisation for Standardization, 1987.

ISO/IEC 9126 (Standard). Information Technology - Software Product Evaluation, Quality Characteristics and Guidelines for their Use. International Organisation for Standardization, 1991.

Jackson, M.: "Problems and Requirements" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 2 - 8.

Jackson, M. Zave, P.: "Domain Descriptions" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 56 - 64.

Jacobson, I.: Object-Oriented Software Engineering. A Use Case Driven Approach. Reading, Mass, 1992.

Jacquemin, C.; Mariani, J.; Paroubek, P. (eds.): Proceedings of the LREC2000 satellite workshop, Using Evaluation whitin HLT Programs: Results and Trends, Athens, Greece, May 30th 2000

JEIDA: A Japanese View of Machine Translation in the Light of the Considerations and Recommendations Reported by ALPAC, USA. Japan Electronic Industry Development Association, Tokyo, 1989.

Johnson, H.; Johnson, P.: "Designers-identified Requirements for Tools to Support Task Analyses" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 259 - 264.

Kade, O.: "Kommunikationswissenschaftliche Probleme der Translation" in: Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt (1968), 1981, pp. 199 -218.

Kapp, V. (ed.): Übersetzer und Dolmetscher. Theoretische Grundlagen, Ausbildung Berufspraxis. Heidelberg, 1974, pp. 174 - 185.

Kampf, M.: "Die Veränderungen der Konsumgewohnheiten und ihre Auswirkungen auf die Nahrungsmittelindustrie" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991[2], pp. 163 - 180.

Karat, C.M.: "Cost-Benefit Analysis of Iterative Usability Testing" in Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 351 - 356.

Keeney, R.L.: Siting Energy Facilities. New York, 1980.

Keeney, R.; Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge, 1993[3]

Kempen, Gerard; Vosse, Theo: "A Language-Sensitive Text Editor for Dutch" in: P. Holt; N. Williams(eds.): Proceedings of the Computers & Writing III Conference, Edinburgh, April, 1990.

King, M. (ed.): Machine Translation Today. Edinburgh Information Technology Series 2. Edinburgh, 1987.

King, M.; Falkedal, K.: "Using Test Suites in the Evaluation of Machine Translation Systems" in : Proceedings of the 13th International Conference on Computational Linguistics (COLING), Hesinki, 1990.

Kiraly, D.C.: Toward a Systematic Approach to Translation Skills Instruction. PhD Diss. Urbana, Illinois, 1990.

Kitchenham, B.:"Towards a Constructive Quality Model. Part I: Software Quality Modelling, Measurement and Prediction." in Software Engineering Journal, July, 1987, pp. 105 - 113.

Kohrt, M.; Küper, Ch.: Probleme der Übersetzungswissenschaft. Berlin, 1991.

Koller, W.: "Anmerkungen zu Definitionen des Übersetzungs"vorgangs" und zur Übersetzungskritik (1974)" in: Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt, 1981, pp. 263 - 274.

Komissarov, V.N.: "Linguistische Modelle (1972)" in: Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt, 1981, pp. 171 - 185.

Krebs, H.: "Anforderungsspezifikation als Grundlage für Softwareanalyse und -test" in: Meckelburg, H-J; Jansen, H. (eds.): Entwicklung und Prüfung sicherheitsbezogener Systeme: Software- und Systemaspekte. Berlin/Offenbach, 1990, pp. 45 - 59.

Krings, H.P.: Was in den Köpfen von Übersetzern vorgeht. Eine empirische Untersuchung der Struktur des Übersetzungsprozesses anhand fortgeschrittenen Französischlernern. Tübingen, 1986.

Kugler, M.; Höge, M.; Heyer, G.; Kese, R.; Kleist-Retzow, B.; Winkelmann, G.: "The Translator's Workbench - an Environment for Multi-Lingual Text Processing and Translation" in: ESPRIT '91 Proceedings of the Annual ESPRIT Conference Brussels. Commission of the European Communities Directorate-General Telecommunications, Information Industries and Innovation (ed.), 1991.

Kugler, M.; Ahmad, K.; Thurmair, G.: (eds.): Translator's Workbench. Tools and Terminology for Translation and Text Processing. Research Reports ESPRIT Project 2315 TWB Volume 1, Brussels-Luxembourg, 1995.

Kußmaul, P.: "Creativity in the Translation Process: Empirical Approaches" in: Van Leuven-Zwart, K.M.; Naaijkens, T. (eds.): Translation Studies: The State of

the Art. Proceedings of the First James S Holmes Symposium on Translation Studies. Amsterdam, Atlanta, 1991, p.91 - 101.

Kuwana, E.; Herbsleb, J.D.: "Representing Knowledge in Requirements Engineering: An Empirical Study of what Software Engineers Need to Know" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 273 - 276.

Lamsweerde, A. van; Darimont, R.; Massonet, P.: "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 194 - 203.

Larson, M.L. (ed.): Translation: Theory and Practice, Tension and Interdependence. Amsterdam, 1991.

Le-Hong, K.; Höge, M.; Hohmann, A.: "User's Point of View of the Translator's Workbench" in: Translating and the Computer 14. Quality Standards and the Implementation of Technology in Translation. ASLIB, 10-11 November 1992, London, pp. 25 - 31.

Lehnert, W.; Sundheim, B.: "A Performance Evaluation of Text-Analysis Technologies" in: AI Magazine Vol. 13, No. 3, 1991, pp. 81 - 94.

Lehrberger, J.: Bourbeau, L.: Linguistic Characteristics of MT Systems and General Methdology of Evaluation. Amsterdam, 1988.

Leimer, H.W.: "Vernetztes Denken im Schweizerischen Bankverein" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 41 - 65.

Leite, J.C.S.P.; de Pádua Albuquerque Oliveira, A.: "A Client Oriented Requirements Baseline" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 108 - 115.

Levin, H.M.: "Cost-Effectiveness Analysis in Evaluation Research" in: Guttentag. M.; Struening, E. L. (eds.): Handbook of Evaluation Research. Vol. 2. Beverly Hills, London, 1975, p. 89 - 122.

Lewis, J.R.; Henry. S.C.; Mack, R.L.: "Integrated Office Software Benchmarks: A Case Study" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 337 - 343.

Lörscher, W.: Translation Performance, Translation Process and Translation Strategies. A Psycholinguistic Investigation. Tübingen, 1991.

LRE/EAGLES - Abstracts from the Technical Annex, Brussels, 1993.

Lubars, M.; Potts, C.; Richter, C.: "A Review of the state of the Practice in Requirements Modelling" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 2 - 14.

Macfarlane, I.A.; Reilly, I.: "Requirements Traceability in an Integrated Development Environment" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 116 - 123.

Mayer, R.: Ein Rechnerunterstütztes System für die technische Dokumentation und Übersetzung. Berlin, Heidelberg, 1993.

MCCall, J.A.; Richards, P.K.; Walters, G.F.: "Concepts and Definitions of Software Quality Factors" in: Software Quality Vol. 1 Springfield, 1977.

Meckelburg, H-J; Jansen, H. (eds.): Entwicklung und Prüfung sicherheitsbezogener Systeme: Software- und Systemaspekte. Berlin/Offenbach, 1990.

Meister, P.: Vernetztes Denken bei der Markteinführung neuer Produkte Dargestellt am Beispiel der Hilti AG" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 145 - 161.

Melby, A.; Warner, T.: The Possibility of Language. A Discussion of the Nature of Language, with Implications for Human and Machine Translation. Amsterdam, 1995.

Meyer, R.: Entscheidungstheorie: ein Lehr und Arbeitsbuch. Wiesbaden, 1999.

Miller, E.F.: "Quality Managment Technology: Practical Applications" in: Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984, pp. 255 - 266.

Miller, E; Howden, W.E (eds.): Intorial: Software Testing and Validation Techniques. IEEE, 1981.

Minaev, Yu. N.; Reshetnyak, Yu. V.: "Constructing an Evaluation Scale in Software Evaluation" in: Measurement Techniques, 30 (1987), 8, pp. 728-730.

MLAP '93 Call for Proposals. Commission of the European Communities, DG XIII, Luxembourg, 1993.

MLAP '94 Call for Proposals. Commission of the European Communities, DG XIII, EC OJC of 15 March, Luxembourg, 1993.

Moll, T.; Ulich, E.: "Einige methodische Fragen in der Ananlyse von Mensch-Computer Interaktion" in: Zeitschrift für Arbeitswissenschaft, 42 (1988) 2, pp. 70-76.

Murine, G.E.; Carpenter, C.L.: "Applying Software Quality Metrics" in: Proceedings from the ASQC Quality Congress, Transactions. Boston, 1983.

Murine, G.E.: "Using Software Quality Metrics as a Tool for Independent Verification and Validation" in: Fifth Annual Int. Phoenix Conference on Computers and Communication '86, Conference Proceedings, Scottsdale, USA, March 26-28, 1986 (1986), pp. 433-437.

Musa, J.D:; Iannino, A.; Okumoto, K.: Software Reliability. Measurement, Prediction, Application. New York, 1987.

Myers, G.J.: The Art of Software Testing. New York, 1979.

Nnagji, B.O.: "Evaluation Methodology for Performance and System Ecomonics for Robotic Devices" in: Computers and Industrial Engineering 14, (1988) 1, pp. 27-39.

Neal, A.S.; Simons, R.M.: "Playback: A Method for Evaluating the Usability of Software and its Documentation" in: Proceedings of the Anniversary Meeting

1985, User Friendly Computing, Zürich, ch, September 23-27, 1985, 2 (1985), pp. 1051-1075.

Neubert, A.: "Pragmatische Aspekte der Übersetzung." in: Grundfragen der Übersetzungswissenschaft, Beihefte zur Zeitschrift Fremdsprachen II, Leipzig, 1968, pp. 21 - 33.

Nida, E.A.: "Science of Translation" in: Language 45, 1969, 483 - 498.

Nida, E.A.; Taber, Ch. R.: The Theory and Practice of Translation. Leiden, 1969, reprinted 1982.

Nord, Ch.: Text Analysis in Translation. Theory, Methodology, and Didactic Application of a Model for Translation-Oriented Text Analysis. Amsterdam, Atlanta, 1991.

Nunnally, J.C.: "The Study of Change in Evaluation Research: Principles Concerning Measurement, Experimental Design, and Analysis" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 101 - 137.

Nunnally, J.C.; Durham, R.L.: "Validity, Reliability, and Special Problems of Measurement in Evaluation Research" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 289 - 352.

Nunnally, J.C.; Wilson, W.: "Method and Theory for Developing Measures in Evaluation Research" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 227 - 288.

Oppermann, R. ; Murcher, B.; Pateau, M.; Pieper, M.; Simm, H.; Stellmacher, J.: Evaluation von Dialogsystemen. Der software-ergonomische Leitfaden EVADIS. Berlin, 1988.

Osterweil, L.: "Integrating the Testing, Analysis and Debugging of Programs" in: Hausen, H.L. (ed.): Software Validation. Proc. Symp. Software Validation. North-Holland, 1984, pp. 73 - 93.

Partsch, H.: Requirements Engineering. Handbuch der Informatik Bd. 5.5. München/Wien/Oldenbourg, 1991.

Piaget, J.: La psychologie de l'intelligence. Paris, 1947.

Pofahl, E.: "Werkzeuge für die Prüfung einer Software" in: Meckelburg, H-J; Jansen, H. (eds.): Entwicklung und Prüfung sicherheitsbezogener Systeme: Software- und Systemaspekte. Berlin/Offenbach, 1990, pp. 107 - 138.

Potts, C.: "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 128 - 130.

Pöyhönen, M; Hämäläinen, R.P.: On the Convergence of Multiattributre Weighting Methods. Helsinki University of Technology Systems Analysis Laboratory. Research Reports, A66, April 1997.

Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991².

Probst, G. J. B.; Gomez, P.: "Die Methodik des vernetzten Denkens zur Lösung komplexer Probleme" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 5 -20.

Rapoport, A. (ed.): Desicion Theory and Decision Behaviour. Basingstoke, 1998

Regnell, B.; Kimbler, K.; Wesslén, A.: "Improving the Use Case Driven Approach to Requirements Engineering" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 40 - 47.

Reinke, U.: "Zur Leistungsfähigkeit integrierter Übersetzungsswysteme" In: Lebende Sprachen 3/94. Bonn, 1994.

Reiß, K.: "Übersetzungstheorien und ihre Relevanz für die Praxis" in: Lebende Sprachen 31 (1), 1986, pp. 1 - 5.

Reiß, K; Vermeer, H.J.: Grundlegung einer allgemeinen Translationstheorie. Tübingen, 1991.

Rosenthal, M.: "Careful Software Evaluation Increases End User Acceptance" in: Data Management 23 (1985) 9, pp. 33-32.

Rothenberg, J.: "Cost-Benefit Analysis: A Methodological Exposition" in: Guttentag. M.; Struening, E. L. (eds.): Handbook of Evaluation Research. Vol. 2. Beverly Hills, London, 1975, p. 55 - 88.

Rushinek, A.; Rushinek, S.: "Accounting and Auditing Software Evaluation with Knowledge Based Expert Systems: An Empirical Multivariate Model" in: Fourth Annual International Conference on Computers and Communications '85, Conference Proceedings, Scottsdale, USA, March, 20-22, 1985, (1985), pp. 250-254.

Ryan, K.: "The Role of Natural Language in Requirements Engineering" in: IEEE Conference on Requirements Engineering, Colorado Springs, 1993, pp. 240 - 242.

Ryan, K.: "Let's Have More Experimentation in Requirements Engineering" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 66.

Sager, J.C.: Language Engineering and Translation: Consequences of Automation. Amsterdam, 1994.

Sager, J.C.: A Practical Course in Terminology Processing. Amsterdam, 1990.

Schick, F.: Making Choices: A Recasting of Decision Theory. Cambridge, 1997.

Schmidt, M.: "Metriken und ihre Diskussion im Zusammenhang mit Software Qualität" in: GI Softwaretechniken-Trends Heft 3.3 (1983) pp. 20-27.

Schmied, W-S.; Winkler, H.: Software-Qualität. Ausgewählte Methoden und Werkzeuge der Softwareprüfung. Siemens-Schriftenreihe data praxis, München, 1989.

Schüller, T.: Integrierte Übersetzungssysteme. Saarbrücker Studien zu Sprachverarbeitung und Übersetzen, Freigang/Reinke (eds.), Band 1, Saarbrücken, 1995.

Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995.

Sewell, D.R.: "A Plan & Goal based Method for Computer-Human System Design" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 283 - 288.

Sharratt, B.: "Memory-Cognition-Action Tables: a Pragmatic Approach to Analytical Modelling" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990.

Slocum, J. et al.: An Evaluation of METAL: the LRC Machine Translation System. In: Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics, Geneva, 1985, pp, 62-69.

Slocum, J. (ed.): Machine Translation Systems. Studies in Natural Language Processing II Series, Cambridge, 1988.

Sneed, H.: "Software-Testen - State of the Art" in: Software Entwicklungs-Systeme und Werkzeuge, 2 Kolloquium, Esslingen, 8-10. September 1987, (1987), pp. 10.3-1-10.3-6.

Somers, H. (ed.): Terminology, LSP and Translation. Studies in Language Engineering. InHonor of Juan C. Sager. Amsterdam, 1996.

Sommerville, I.: Software Engineering. Reading, Mass. 1996[5].

Spies, C.: Vergleichende Untersuchung von Integrierten Übersetzungssystemen mit Translation-Memory-Komponente. Saarbrücker Studien zu Sprachverarbeitung und Übersetzen, Freigang/Reinke (eds.), Band 3, Saarbrücken, 1995.

Stein, D.: Theoretische Grundlagen der Übersetzungswissenschaft. Tübingen, 1980.

Sternberg, R.J.: "Toward a Triarchic Theory of Human Intelligence" in: The Behavioural and Brain Sciences 7, pp. 269 - 315.

Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975.

Struening, E.L.; Guttentag, M.: "The Handbook: Its Prupose and Organisation" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 3 - 9.

Sullivan, J.L.; Niemi, R.G.(eds.): Sage University Series on Quantitative Applications in the Social Sciences. Beverly Hills and London, 1982.

Sundermeier, B.; Versteeg. P.: "Software-Prüfanleitung für Projektleiter. Wie man bei Minimierung der Projektkosten qualitativ hochwertige Software produziert" in: Meckelburg, H-J; Jansen, H. (eds.): Entwicklung und Prüfung sicherheitsbezogener Systeme: Software- und Systemaspekte. Berlin/Offenbach, 1990, pp. 91 - 105.

Teichroew, D.; Hershey, E.A.: "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems" in: IEEE Transactions on Software Engineering vol. SE-3 (1), 1977, pp. 41 - 48.

Telematics Programme - Mid Term Review. Comission of the European Communities, July, 1993

Thaller, G.E.: Qualitätsoptimierung der Software-Entwicklung. Das Capability Maturity Model (CMM). Braunschweig/Wiesbaden, 1993.

Thaller, G.E.: Verifikation und Validation. Software Tests für Studenten und Praktiker. Wiesbaden, 1994.

Thiel, G.: "Ansätze zu einer Methodologie der übersetzungsrelevanten Textanalyse" in: Kapp, V. (ed.): Übersetzer und Dolmetscher. Theoretische Grundlagen, Ausbildung Berufspraxis. Heidelberg, 1974, pp. 174 - 185.

Thompson, H.: 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, 1992.

Toury, G.: Descriptive Translation Studies - and beyond. Amsterdam, 1995.

Twain, D.: "Developing and Implementing a Research Strategy" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 27 - 52.

Van Leuven-Zwart, K.M.; Naaijkens, T. (eds.): Translation Studies: The State of the Art. Proceedings of the First James S Holmes Symposium on Translation Studies. Amsterdam, Atlanta, 1991.

Vainio-Larsson, A.; Orring, R.: "Evaluating the Usability of User Interfaces: Research in Practice" in: Diaper, D.; Gilmore, D.; Cockton, G.; Shackel, B. (eds.): Human Computer Interaction - INTERACT '90, Elsevier, IFIP, 1990, pp. 323 - 328.

Vasconcellos, M. (ed.) Technology as Translation Strategy. Amsterdam, 1988.

Vernay, H.: "Elemente einer Übersetzungswissenschaft (1974)" in: Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt, 1981, pp. 236 - 249.

Vetter, M.: Strategie der Anwendungssoftware Entwicklung. Methoden, Techniken, Tools einer ganzheitlichen, objektorientierten Vorgehensweise. Stuttgart 1993[3].

Vick, C.R.; Ramamoorthy, C.V (eds.): Handbook of Software Engineering. New York, 1984.

Vogelin, C.F.: "Multiple Stage Translation" in: International Journal of American Linguistics 20, pp. 271 - 280, 1954.

Weiss, C.H.: "Evaluation Reserach in the Political Context" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 13 - 26.

Weiss, C.H.: "Interviewing in Evaluation Research" in: Struening, E.L.; Guttentag, M. (eds.): Handbook of Evaluation Research. Vol. 1. Beverly Hills, London, 1975, p. 355 - 395.

WHA: Word House Associates: Trados Translator's Workbench II versus IBM Tranlsation Manager/2. Baambrugge 1993.

Wilson, M.: "Task Models for Knowledge Elicitation" in: Diaper, D. (ed.) Knowledge Elicitation: Principles, Techniques, Applications, 1989, pp. 195 - 219.

Wilss, W.: Übersetzungswissenschaft. Probleme und Methoden. Stuttgart, 1977.

Wilss, W. (ed.): Übersetzungswissenschaft. Darmstadt, 1981.

Wilss, W.: The Science of Translation. Problems and Methods. Tübingen, 1982.

Wilss, W.: Kognition und Übersetzen. Zu Theorie und Praxis der menschlichen und der maschinellen Übersetzung. Tübingen, 1988.

Wilss, W.: Übersetzungsfertigkeit: Annäherung an einen komplexen übersetzungspraktischen Begriff. Tübingen, 1992.

Wilss, W.: Knowledge amd Skills in Translator Behaviour. Amsterdam 1996.

Winterfeldt, D. von; Edwards, W.: Decision Analysis and Behavioral Research. Cambridge, 1986.

Woods, W.A.: "Progress in NLU - An Application to Lunar Geology"; in: AFIPS Conference Proceedings 42, 1973, pp. 441 - 450.

Wright, S.E.; Wright, L.D. Jr.: Scientific and Technical Translation, Amsterdam, 1993.

XIII Magazine, May 1993 Issue No.10, Brussels, 1993.

Yeh; R.T.: "Requirements Analysis - a Management Perspective" in: Proceedings of the COMPSAC '82, Chicago, 1982, pp. 410 - 416.

Yourdon, E.: Modern Structured Analysis. Englewood Cliffs, 1989.

Zave, P.: "Classification of Research Efforts in Requirements Engineering" in: Second IEEE International Symposium on Requirements Engineering, Los Alamitos, California, 1995, pp. 214 - 216.

Zimmermann, T.P.: "Vernetztes Denken in einer Werbeagentur" in: Probst, G. J. B.; Gomez, P. (eds.) Vernetztes Denken. Ganzheitliches Führen in der Praxis. 1991², pp. 81 - 106.