PaTerm: A Case Study in Information Mapping

Steven Verjans & Niels Ole Bernsen

Abstract

Today's human-computer interface designers are able to select from a rapidly increasing number of input/output modalities for a given application. This raises the problem of how to select the right combination of modalities for the application in a way which is principled rather than *ad hoc*. The present paper explores a methodology for this purpose, i.e. the Information Mapping Methodology for performing a mapping between task domain information and appropriate input/output modality combinations. The Information Mapping methodology is applied to PaTerm, an interactive tool for adding lexical databases to the Machine Translation system PaTrans.

Keywords: Multimodal Systems, Interface Design, HCI Design Methodology, Information Mapping, Modality Theory.

1. Introduction

Literally thousands of different combinations of input and/or output (information) *representational modalities* are currently becoming available to designers of interfaces for human-computer interaction, from unimodal spoken language input to complete multimodal virtual reality interactive systems. Each single modality or multimodal combination has its own specific capabilities of representing or conveying information and it is obviously important to be able to select the right combination of modalities for a given application. The question is how this might be done in a principled manner so as to optimise the usability of the interface, given the specific purpose of the artifact to be designed. Answering this question involves addressing the research agenda of Modality Theory whose development forms part of the ESPRIT Basic Research project AMODEUS II (Barnard 1993). The agenda is as follows (Bernsen 1993a):

(1) to establish sound foundations, both conceptually and in terms of an operational taxonomy, for describing and analysing any particular type of unimodal or multimodal output representation relevant to human-computer interaction (HCI);

(2) to create a conceptual framework for describing and analysing interactive computer interfaces so as to cover both input and output of information;

(3) to apply the results of steps (1) and (2) above to the analysis of the problems of information-mapping and information transformation between work/task domains and human-computer interfaces in information systems design.

Briefly, the main problem raised by Agenda Item 1 is how to build a principled foundation for addressing the information representing capabilities of thousands of different, potentially useful combinations of output modalities. The only viable approach seems to be through the definition and analysis of a limited set of elementary or *unimodal* modalities from which any particular modality combination can be built. Results from adopting this approach are reported in (Bernsen 1994)**.** We have just begun to address Agenda Item 2. Agenda Item 3 is to develop an operational 'bridging' representation between the science base of Modality Theory and design practice (Barnard 1991). We call this bridging representation the *Information Mapping Methodology.* This paper explores the Information Mapping Methodology proposed in (Bernsen and Bertels 1993) through applying the methodology to PaTerm, an interactive tool for adding lexical databases to the Machine Translation (MT) system PaTrans. The information mapping study reported here is the first comprehensive study in which the IMAP Methodology has been applied to a full-scale realistic design process.

The plan of the paper is as follows: Section 2 provides a brief description of the Information Mapping Methodology. Section 3 presents PaTrans and the PaTerm tool. Section 4 describes our work with the PaTerm designers and the results of Steps 1-3 of the Information Mapping Methodology (IMAP). Section 5 presents the information mapping and trade-off steps (Steps 4 and 5) of the methodology. Preliminary results and lessons learned from the present case study are discussed in Section 6.

2. The Information Mapping Methodology

The methodology proceeds in five steps:

Step 1: Identification of Information and Tasks
The first problem is to identify the information to be exchanged by user and system during task performance in the application domain. So the aim of Step 1 is to obtain the information from the task domain which is needed to select a reasonable and possibly optimal mapping from task domain information to interface input/output representation. The nature and variety of the information relevant to this end should not be underestimated. A central part of the information needed for solving an information-mapping problem is information on users' tasks. However, any reasonably versatile or powerful IT artifact can be used for performing a multitude of different tasks and it is obviously not always possible during practical systems design to consider each and every such task as to its information-mapping requirements. In other words, it will normally be necessary during practical interface design to be selective as to the tasks to be considered in any detail. The ideal way to be selective is to identify a limited set of *representative* tasks or scenarios to be performed by using the intended artifact and carry out the information-mapping analysis on these. The problem, of course, is that no guaranteed method for generating an appropriate set of representative scenarios currently exists in HCI or usability engineering. Let us simply assume at this stage that the best current methods or

heuristics are being applied for the purpose of identifying representative tasks (cf. Bernsen and Bertels 1993).

The results of Step 1 would normally be (a) high-level information required to solve the information-mapping problem and (b) a small set of representative tasks which users should be able to carry out on or with the intended artifact. These results constitute an operationalisation of the information-mapping problem. Step 1 is crucial to the success of the methodology. To the extent that the selected tasks lack representativity, the risk is that requirements important to information mapping have been overlooked.

Step 2: Selective Task Analysis

In Step 2 the selected representative tasks are individually analysed in as much detail as possible in order to identify their goals and initial states, their preconditions, the activities and procedures involved, how they might go wrong, the task (work) environment, the intended users and their experience, etc. The analysis should primarily aim at revealing the input/output information representation needs of the tasks. That is, while a more or less complete task analysis may be performed either formally or informally, not all of the information it produces needs to be explicitly represented in order that IMAP will be successful. Step 2 is closely related to Step 3:

Step 3: Information Representation

In Step 3 the relevant information acquired through Steps 1 and 2 is represented explicitly and succinctly, for instance using the Design Space Development (DSD) notation for representing design space structure (see below). In principle, this representation should contain everything which is relevant to the input/output modality choices to be subsequently made. The representation should preferably be expressed in terms of Modality Theory in order to facilitate the mapping of information from the requirements analysis onto input/output modalities. Step 3 makes explicit the requirements on interactive information to be satisfied by the interface to be designed and concludes the first main phase of IMAP.

Step 4: Information Mapping

Step 4 consists in applying Modality Theory to the results of Steps 1 to 3 above in order to map the collected task domain information onto a suitable set of input/output modalities. From the point of view of IMAP, Modality Theory consists in a large set of rules and rule-like knowledge, such as, e.g., the following rules:

> Visualise specific information in 1D, 2D or 3D spatial, temporal development being important to the visualisation <->
>
> Consider using dynamic analogue graphics.

> Visualise specific information such that freedom of visual inspection is less important than development, movement or change <->

Consider using dynamic analogue graphics.

An application of Modality Theory for the purpose of information mapping can be thought of as applying rules such as those just illustrated, assuming, that is, that the results of Step 3 of IMAP have been expressed appropriately for the purpose of applying Modality Theory to the design of this particular artifact, and that Modality Theory includes knowledge which is relevant to deciding which input/output modalities to use for this artifact. The 'rule-like knowledge' mentioned is knowledge embodied in Modality Theory which has not been explicitly expressed as rules but which nevertheless supports information mapping. The result of information mapping will be sets of possible input/output modalities and modality combinations which are capable of representing the information needed for the representative tasks. It seems likely that information mapping will often produce several alternative solutions which subsequently have to be compared and traded off against one another.

Step 5: Trade-Offs

In Step 5 a 'higher level filtering' is performed to trade off potential solutions against one another given the results of Steps 1 through 4. The trade-off process may be explicitly represented in some form of Design Rationale representation (see below). The result of Step 5 is a solution to the task domain/interface mapping problem together with its Design Rationale. In some cases, several solutions can be expected to emerge from the trade-off process with identical ‘scores’ or suitability evaluations.

The five steps of the Information Mapping methodology described above can be clearly separated for analytical purposes. In practice, however, iteration and backtracking should be expected. The information initially collected and represented in Steps 1 to 3 is not necessarily sufficient for carrying out a complete information mapping process (Steps 4 through 5). The application of Modality Theory may raise additional questions about all aspects of the artifact to be designed, which can only be answered by further analysis of tasks, task domains, intended users and other constraints on the design process. It should further be remembered that Modality Theory so far only covers output modalities. When we appeal to properties of input modalities in what follows, these properties have not been similarly grounded in principled theory. It is also important to note that we studied PaTerm *after* its completion rather than during its design or implementation phase. This approach was dictated on practical grounds and raises particular difficulties of collecting information relevant to information mapping, which would not arise in using IMAP during normal design practice.

3. Patrans and Paterm

PaTerm and PaTrans were developed at CST, the Centre for Language Technology in Copenhagen and have been sold to LingTech, a Danish translation company specialised in patent text translation. PaTrans is itself based on the European MT project Eurotra. The Patrans

MT system was originally developed by CST to handle patent texts in the petrochemical industry and to translate those texts from English to Danish. The idea of the PaTerm tool is to allow the customer

LingTech to extend the coverage of PaTrans to many other technical areas for which patent texts are to be translated from English to Danish.

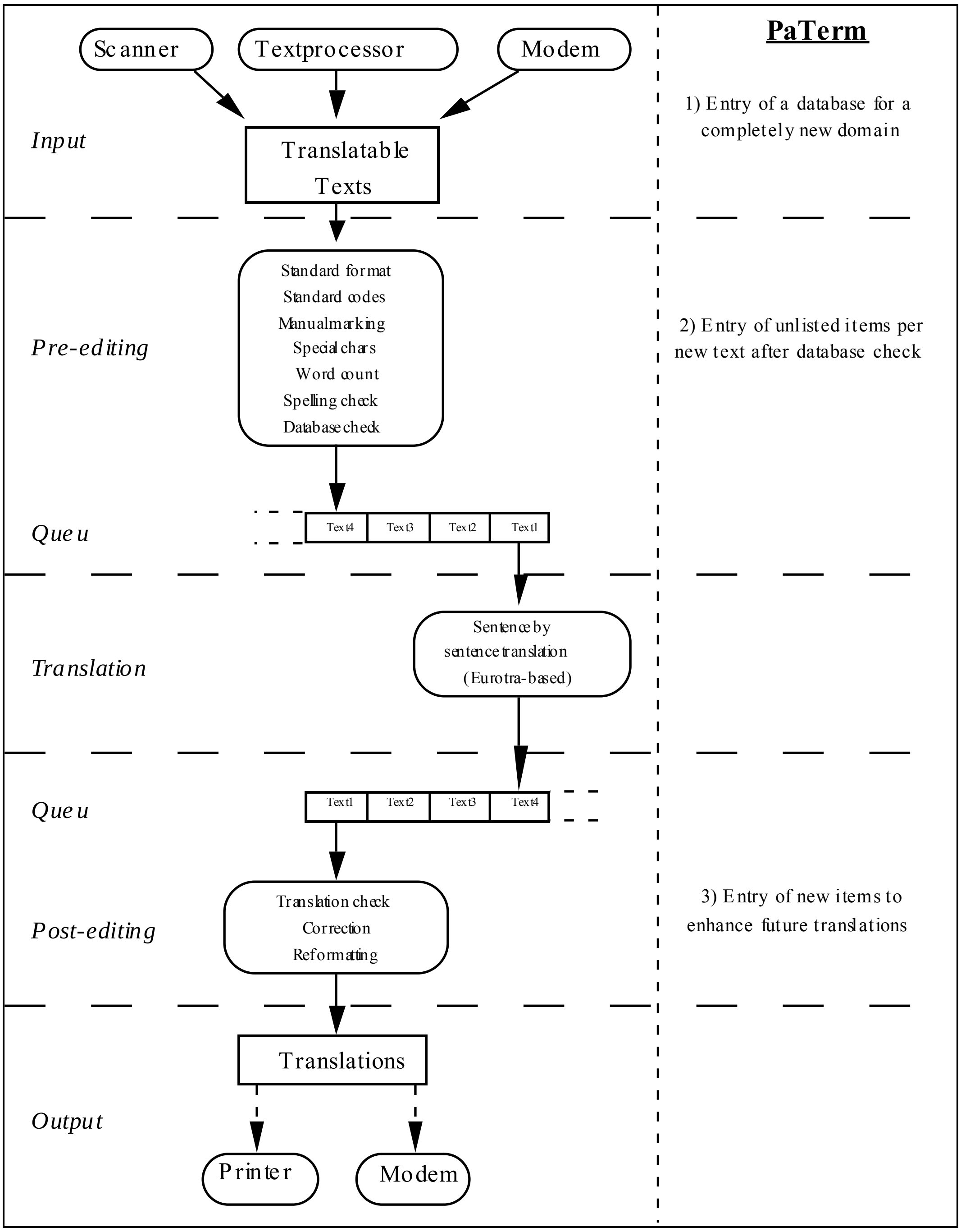


Figure 1. The complete translation process. The right-hand side shows at which stages of the translation PaTerm can be used. The left-hand side shows the translation process in chronological order (from top to bottom).

In order to analyse PaTerm, we first have to describe the complete translation process. This process is diagrammatically represented in Fig. 1.

The *first step* in text translation is the setting up of the translation system PaTrans. PaTrans consists of multiple modules, of which the grammar and the dictionaries are the most important ones from our point of view. PaTrans uses several different dictionaries to translate a text. It has a general dictionary (GenD) which covers frequent English words and constructs. For the petrochemical domain a general chemical (ChemD) dictionary as well as a petrochemical dictionary (PetroD) have been added. ChemD and PetroD were drafted from a large corpus of petrochemical patent texts. The decision to use two different dictionaries for these terms was made by CST. It was discovered that some of the terms found in the corpus were more general than others and could be re-used in task domains other than petrochemistry. The order in which the dictionaries are to be searched during translation is specified for each text separately. This is done through adding, during the pre-editing phase, a special code at the top of the text.

The first possible use of PaTerm is to enter the database(s) for a new patent text domain (cf. Fig. 1). This was not done for petrochemistry, however, because PaTerm did not exist at the time and as the initial dictionaries ChemD and PetroD were built by the linguists at CST rather than by the Lingtech end-user staff.

The *second step* in the translation process is to collect texts for translation. This can be done through standard methods for creating or obtaining text in electronic form (using modem, scanner, word processor, FTP, e-mail, etc.).

The collected texts must then be pre-edited, which is the *third step* in the translation process and the second stage where PaTerm can be used (cf. Fig. 1). Pre-editing consists in the following steps:

1) Firstly, the original text is converted into a standard format which can be read by PaTrans. In this process, part of the layout is lost while some layout is coded.

2) Then, standard codes are added, such as a text number, a client number, the order in which the dictionaries have to be scanned, etc.

3) Special parts of the text, such as tables, graphs or special characters are manually marked as 'not to be translated'. Very long sentences are split by special marks to enable translation.

4) Finally, a word frequency table is created; all words in the table are spell-checked and then looked up in the dictionaries. The result is a table of the following form, where the first column indicates the frequency of an item, the second column the number of the dictionary in which the element was found, and the third column the element itself:

1	-	'marked'
1	+	aberted
21	0	the
10	0	a
2	2	lubricating_oil

The second column is particularly important for the user of PaTerm, since it indicates whether or not an item has to be entered in a dictionary. '-' indicates that the item was not found in any dictionary and could not be morphologically decomposed by the grammar, whereas '+' indicates that the unfound item was preliminarily decomposed. The word 'aberted', for instance, is a misspelled version of 'aborted'. The grammar recognised the '-ed' ending as a verbal suffix, which allowed it to decompose the word into 'abert-ed'.

As a *fourth step* in the translation process, the pre-edited texts are put in a queue to be automatically translated by PaTrans. The translated texts are put in another queue for post-editing (cf. Fig. 1). Some extra information, such as the CPU-time that was needed for translation, is added to the text.

Post-editing, the *fifth step* in the translation process, is for the greater part a manual task. It consists in:

1) Translation checking.
2) Correction of translation.
3) Reformatting the layout of the original text.

This is the third and last stage where PaTerm can be used. When checking the translation, the post-editor may want to add or change terms in the dictionary to improve future translations.

Finally, as a *sixth step*, the post-edited translation may be output by printer, modem, fax, e-mail, FTP, etc.

4. Approch and Results of the First Information Mapping Phase

4.1 Our approach to the analysis of PaTerm

Steps 1 to 3 are closely related and may in practice form an integrated process. The acquisition of information happens iteratively. Our approach was the following:

a) We started by having a demonstration by the designers of the completed artifact. This provided us with a rough idea of the information that is to be exchanged between the system and the user. The result was high-level information about initial situation and goal of the artifact, and distinction among a number of representative tasks. This corresponds to Step 1 of IMAP.

b) Next, we tried to shape the collected information in the DSD representation (Bernsen 1993b, 1993c), i.e., Step 3 of IMAP, but it soon turned out that we did not have enough information. We decided that we needed more information (return to Steps 1 and 2) which was preferably to be obtained from a list of directed questions developed on the basis of the information already available.

c) We generated questions through in-depth analysis of what we then considered to be the representative task, namely the entry into a dictionary of new lexical terms and their translations, corresponding to Step 2 of IMAP. We performed this detailed analysis by asking very simple, practical questions from the point of view of the user. After a first, informal brainstorming we (semi-) formalised the questions to build a list of specific questions for the next interview. The formulation of the questions resembles the DR/QOC-notation (MacLean et al. 1991, 1993) but is less formal. We imported the idea of formulating Questions (or problems) and Options but did not use the DR/QOC notation for Criteria and how they relate to Options (see Fig. 2). At the same time we tried to locate the problems in the design space through asking questions such as: is this a problem for the system to solve? Is it an organisational problem? Does the user need much experience to solve the problem? Etc.

Problem: *PTO*	**Options**	**Evaluation**
How will the $TextT_n$ be searched for lexical items? How will LIST, the list of all items that will eventually be entered in TaskD, be drafted?	(1) An automatic search system, comparing words in $TextT_n$ with words that are already in GenD, ChemD and PetroD. (2) Manually scanning of texts, looking for new items. (3) Human-controlled automatic process.	- Option 1 will not suffice for various reasons (problems 3-5 in Appendix 1). - Option 2 seems to be impossible from a practical point of view. - Option 3 seems to be the only plausible one, for reasons of efficiency and completeness.

Figure 2. Semi-formal problem notation. The leftmost cell contains the problem, the center cell lists several possible options, and the rightmost cell provides a short evaluation of the options. The code in italics on the first line of the left-hand cell denotes the location of the problem in the design space. PTO stands for Pre-Task Organisation.

d) Using the described procedure we formulated 30 odd questions. We sequentially broke down the problems into sub-problems, until we were left with 'atomic problems'. This approach proved very useful to the analysis. It allowed us to achieve a high degree of detail in the questions we asked in the second interview. In this interview we had answers to these questions, which - naturally - forced changes in our initial interpretation of some of the information. Appendix 1 contains all the questions and answers of the second interview.

e) The collected information was then represented in the DSD notation, corresponding to Step 3 of IMAP (see Appendix 2: DSD No. (1)).

f) Again it turned out that the information we had was neither complete nor accurate enough for the actual information mapping, so we had to backtrack to Steps 1 and 2 of IMAP. We went back to CST for a third interview, from which we gathered enough information to create DSD No. (2), which is shown in Sect. 4.4 below.

4.2 IMAP Step 1: High-level information on PaTerm

(a) High-level task domain information:

Initial situation:

- A machine translation system PaTrans, which uses two types of database or dictionary. A general database containing general lexical items (GenD) and one or more - possibly empty - task domain databases (TaskD). In addition, PaTrans contains a (formal) grammar (GRAM, cf. Fig. 4 below), which uses the database items during translation. PaTrans runs under X Windows™ on UNIX machines.

- A client company (CLIENT) which wants to have patent texts translated in its domains of expertise. These texts will be called TextT_n.

- A translation company (TRANS), which will create lexical database(s) for each new task domain.

- An organisational decision made by TRANS: there will be one or more working copies of each TaskD, in which users enter new lexical items, as well as an authorised copy of TaskD, which will be thoroughly checked by another user called the ADMINISTRATOR.

Design Goal:

- Develop an interactive, easy-to-use tool for coding (entering) new lexical items. It will have to run under X-Windows™ on UNIX-machines.

- The tool will be used to code into TaskD English TERMS, syntactic information on the English terms, the Danish translation of the terms and syntactic information on the Danish terms.

- The tool will be used in three different situations: (1) to make a completely new dictionary; (2) to add items to a dictionary before translation, and (3) to add/change items after the post-editing of translated texts.

- The design specifications in the PaTrans contract state that a user interface is to be developed to allow non-linguists to enter lexical items in the dictionaries.

(b) The tasks to be done by using PaTerm. As the overall number of tasks for the tool is limited, they are all listed below:

- information entry (coding);

- information change;

- dictionary search;
- dictionary management/maintenance;
- on-line help.

As the contract stipulated, the main task of PaTerm is the entry of lexical items. That is the main reason why we studied the information entry more closely in Steps 2 to 5. We may address some of the other tasks briefly when needed. Another reason for concentrating on the lexical entry task is the fact that it may involve dictionary search and on-line help.

4.3 IMAP Step 2: Representative task analysis

	Situation (1): Create new dictionary	Situation (2): Add unlisted items	Situation (3): Add/change items after post-editing
Pre-Task Organisation:			
1 List of items to be entered	Available	Available	NA
2 Translation of items	Available	NA	NA
3 Text to be translated	Available	Available	Available
4 Translated text	Available	NA	Available
User Task:			
5 Enter item in dictionary	Yes	Yes	Yes
6 Search dictionary and display homographs	Yes/Optional	No/Optional	Yes/Optional
7 Allow use of existing item as template	Yes/Optional	No/Optional	Yes/Optional
8 Find grammatical information	Yes	Yes	Yes
9 Enter grammatical information	Yes	Yes	Yes
10 Find trans(item)	No	Yes	Yes
11 Enter trans(item)	Yes	Yes	Yes
12 Find grammatical information for trans(item)	Yes	Yes	Yes
13 Enter grammatical information for trans(item)	Yes	Yes	Yes
14 Check result of Steps 5-13	Yes	Yes	Yes
15 Compare with existing items	Yes	No/Optional	Yes
16 Save information from Steps 5-15	Yes	Yes	Yes

Table 1. A detailed breakdown of the information coding task which was selected for its representativity of the use of PaTerm.

Repeated application of Step 2 resulted in the following detailed description of the information coding task (see Table 1). The table shows the main parts of the information-coding task in the three situations where PaTerm can be used (cf. Fig. 1). In addition to the three different situations (or stages in the translation process) in which PaTerm can be used, the designers wanted to distinguish between two user levels:

- at the beginner/intermediate level they wanted a step-by-step entry of lexical information, which requires a multiple-screen representation of information;

- at the expert level they wanted a kind of form-filling entry of the lexical information, which may be done using a one-screen representation in which only strictly relevant information is displayed.

All this information is important for the succinct information representation in Step 3 of IMAP.

4.4 IMAP Step 3: Information representation

Repeated application of Steps 1 to 3 resulted in DSD No. (2), a representation that should contain all the information that is relevant to the lexical information coding task:

Design Project: PaTerm, developed at the Center for Language Technology, Copenhagen (completed at the time of analysis).			
DSD No. 2	**Date: 30.5.94**	**Sign: SV/NOB**	
A. General constraints and criteria			
Overall design goal(s)			**DSD No.**
An interactive tool for adding lexical items to the Task Domain dictionaries of the PaTrans patent text translation system.			
General feasibility constraints			
Not known.			
Scientific and technological feasibility constraints			
The tool should run in an X Windows environment on UNIX-machines.			
Design process type			
Commercial.			
Designer preferences			
Use of the OSF/Motif™ design package for X Windows. This implies some restrictions on the representational modalities which are possible. Motif supports most graphical modalities, such as windows, text, images, icons, pictures. Sound may be possible (depending on the computer system), but is not supported by Motif. Animation is possible, but not supported. On static graphics, however, there are hardly any restrictions.			
Realism criteria			
The translation produced by PaTrans (with support of PaTerm) should be of an acceptable quality, i.e. the main structure of texts and most of their technical terms should be translated properly. The translated texts are post-edited by an experienced translator.			
Functionality criteria			
Make sure that the artifact can support the lexical entry task, that it can assist the human in this process, by offering as much support as possible.			
Usability criteria			
Maximise the naturalness of interaction between user and system. Allow flexibility in the coding process. Introduce different user-levels and situation-dependent interaction, to maximise usability.			

B. Constraints and criteria applied to the artifact within the design space

Collaborative aspects

The translation process involves several steps (see Documentation: Fig. 1). Each step may involve a different operator. The cooperation between operators is very important for the success of the translation. The operators can:

1) enter texts on a workstation;
2) pre-edit texts;
3) enter new items in dictionaries;
4) monitor the process of translation by PaTrans;
5) post-edit translations;
6) enter items after post-editing;
7) output translations;
8) manage/ maintain Task Domain dictionaries;
9) manage/ maintain the General dictionary;
10) manage/ maintain the translation system;
11) handle administration and invoicing of translations.

Depending on the cooperation agreements between the people doing these tasks, the PaTerm tool will be used differently.

Organisational aspects

- The Terms to be entered in a Task Domain dictionary are supplied in different ways, depending on the situation (see Documentation: Table 1).
- Depending on the cooperation agreement, one or more possible interpretations and translations of a Term are to be entered. Either the first user (operator 3, cf. the list of operators above) enters all possible translations, or he/she enters only the relevant one (and operator 6 may have to add interpretations later). This has impact on the information that has to be represented to operator 3.
- Changes to GenD, the general Dictionary, can only be made by operator 9, CST.
- Support for PaTerm and PaTrans is provided by CST.
- There may be several working copies of TaskD. These have to be approved by operator 8, called the administrator. This most probably happens after post-editing of the translation, to make sure that the right translation(s) of terms have been entered.
- Users must have access to the original texts ($TextT_n$) and to support material such as specialised dictionaries and grammars.

System aspects

- There is a hierarchy in the use of the dictionaries in PaTrans. This hierarchy is coded in the first line of every text during pre-editing.
- The existing dictionaries can be searched.
- PaTrans uses only a limited amount of grammatical information, which has a fixed order and form. PaTrans needs fixed codes for each piece of grammatical information.
- PaTrans allows multiple entries for each lexical item. During translation, the user can specify if he/she wants only one possible translation, or multiple translations.
- The Term data are internally represented in PaTrans in two different ways. One has multiple indexing for searching and coding purposes. The other has only one index, for fast translation purposes.

Interface aspects

Available media are:
- Keyboard and mouse input.
- Graphical and acoustic output.
Other media are possible (such as touch screen input), but do not seem to have been considered.

Task aspects

System tasks:
- Allow change and correction at any time.
- Provide manuals at any time.
- Provide on-line help.
- Check coded information for completeness and consistency. The system developers chose to solve this problem by directing the information input through a kind of finite state net. This was possible because the number of different pieces of grammatical information is limited, and because each piece of information that is entered limits the number of choices for the next piece. (See Documentation: Fig. 4). This finite state net provides lists of options from which the user is to choose one. This way one can avoid typing errors.

User tasks:

1) Find item to enter in a TaskD.
This task is different for each situation:
- In Situation 1, a list of items is provided by TRANS.
- In Situation 2, the word-list made in the pre-editing phase provides the items to be entered. This word list does not contain multiple word items such as, e.g., lubricating oil, neither does it contain words that are used in an interpretation which is different from the ones that are in GenD or TaskD already. Such cases are left to the operator who performs the post-editing (Situation 3).
- In Situation 3, the task of finding items to enter is very hard. It must be performed by closely checking the translation produced by PaTrans.

2) Enter item in TaskD.
- For this task, correctness is very important. What is to be entered is a sequence of letters that build a word, or more generally, a lexical item of a specific language. The spelling of this lexical item (English in this case) is very important, so immediate feedback on the spelling is necessary. This feedback will have to be graphical, since spelling is done graphically. The feedback needs to be very clear.
- The user is supposed to enter the British English spelling of the lexical item, if possible, for reasons of consistency.
- The way the entry screen should look, depends on the chosen user level. It may be either an empty screen (beginner/intermediate level) or a screen that looks like an empty form that has to be filled in (expert level).

3) Search dictionaries and display homographs.
- This task is optional in all situations, but advisable in Situations 1 and 3, since the problem of homographs is very real in those situations. It is less so in Situation 2 where a dictionary search has happened before the creation of the word list.
- The information that is to be displayed is again linguistic. It consists of lists of lexical items and their translations and grammatical information.
- The different items must be clearly separated from each other.
-The main parts of the lists are the English lexical item and its Danish translation. It may be necessary to mark these items as being important.

(contd.)

(User tasks contd.)
- One may want to mark the differences between the lists so as to get a quick overview of the different interpretations of an English lexical item. This may introduce some feasibility problems or be very time consuming.

4) Allow use of existing items as templates.
- It is possible that a new item resembles an existing one very much. One may want to make a copy of an existing item and then make some minor changes to that copy. This would probably save time. This user task will then resemble task 11.

5) Find grammatical information.
- This may be done by using dictionaries and grammars.
- English words may have multiple grammatical uses (e.g. they may take two or more prepositions, they may be either verb or noun, e.g., RECORD). In that case a problem arises: which of these alternatives to enter? And who will decide which alternatives to enter? This problem does not have a direct influence on the information representation, however. This problem clearly is an organisational one.

6)Enter grammatical information.
- Again it is very important to avoid typing mistakes. This problem has been solved by using a finite state net representation, which allows the system to use lists from which the user can choose. When entering exceptional cases, however, the typing mistake problem may re-appear.
- The information that has to be entered is linguistic. It can be displayed graphically or acoustically. Here the criteria for choosing between media are speed of the information transfer, persistence of the information display and consistency with the overall information display.
- Presuming that the information to choose from has been efficiently represented, the choice of input modalities remains. When choosing between keyboard, mouse, touch screen and voice, the following criteria apply: availability (machine restrictions), feasibility (implementation time constraints), efficiency (speed), user expertise, consistency and user friendliness.
- Feedback on the entry is necessary again, as is the possibility to change or correct entered information. Again, graphical feedback seems to be the best choice (cf.. user task 1).

7) Find trans(item).
- In Situation 1 the translation is supplied in the corpus list (cf. Documentation: Table 1). In Situation 2 and 3 this may pose some real problems:
- does one enter only one translation or several relevant translations or even all possible translations? (Each translation presupposes a new lexical item to be entered);
- if no translation exists, no entry is made and the term remains untranslated. The fail-soft system that is implemented on top of the translation system, will mark the word as 'untranslated' and the sentence in which the word appears as 'containing an untranslated word'. The post-editor must then correct the translation if necessary;
- if the lexical item is used in its English form in Danish (as are many technical terms) the item is to be entered in the dictionary, together with its Danish usage.
- A problem with this task seems to be the lack of specialised English-Danish dictionaries.
- The translation system can handle most special cases, such as lexical items which partly retain their English morphology.

8) Enter trans(item).

9) Find grammatical information for trans(item).

10) Enter grammatical information for trans(item).
For tasks 8 to 10 no special matters of representation arise (cf. tasks 2, 5 and 6).

11) Check result of tasks 1-10.
- The representation of the completed information depends on the user level that has been chosen. At the expert's level the use of the filled-in form seems to be most consistent with the representation throughout the coding task. At that level there is the problem of marking the relevant information in the form. Should the heading of each piece of information be stressed or the information itself?
- At the beginner's level one may want to use the same form as in the expert's level, or choose another representation. It may be most consistent to keep the feedback representation that is used during the actual coding.
(contd.)

(User tasks contd.) 12) Compare with existing items. - In situations 1 and 3 this task seems necessary, since there may be homographs in the dictionaries. The representation can be compared with the one in task 3. - In situation 2 this task can be optional, since in most cases there will not be any homographs. 13) Save the information for tasks 1 to 12. - The system may first have to check the databases for identical entries to avoid exact copies in the dictionaries. - The task of maintaining and managing the dictionary files is not part of the representative task of coding information. One will probably want a completely separate part of the interface to handle this task.
User and user Experience aspects
The three situations in which PaTerm can be used seem to ask for different users: - In Situation 1 the coding task can be performed by anyone who has enough linguistic background in English and Danish to enter the grammatical information. Task Domain knowledge is not needed, since the translations of the lexical items are provided. - Situation 2 demands more task domain knowledge from the user, since he/she will have to look for translation(s) to enter. The linguistic knowledge that is needed remains the same as in Situation 1. - Situation 3 requires a real translator, since small translation nuances may need to be identified and entered. Some task domain knowledge may be needed, but it seems less important at this stage. Maybe the possibility of outside consultancy needs consideration at this stage (but that possibility must always be available in translation). Each of these users is preferably a Danish native speaker with a good knowledge of English. - The representation aspects of the two user levels were discussed above when we discussed the different user tasks. - The need for at least two user levels seems to be clear. One wants to use examples for the coding of grammatical information at the beginner's level, but no longer at the expert's level. Moreover, expertise in the use of codes for the specification of grammatical properties will speed up the entry of lexical items, so a different level of representation is needed here too. - Support should be available at all user levels. There should be both linguistic and technical support. This should be clearly separated from the coding screen. The user must be able to access support information continuously and repeatedly. A static representation, therefore, seems preferable.
C. Hypothetical issues
- Do specialised dictionaries and grammars (English - Danish) exist? The answer seems to be NO. - Should consultancy be provided? - Should novel users have some training in using PaTerm? - Can touch screen or voice be useful as input modalities?
D. Documentation
- Figure 1: The complete translation process. Especially the three situations in which PaTerm can be used, should be noted. - Figure 4: Part of the finite state net used to enter grammatical properties. - Table 1: Detailed description of the representative task in the three situations of use. - Appendix 3 Figs. A and B: The current PaTerm displays.
E. Key: DSD No. (n) indicates the number of the current DSD frame. 'Null' means that the artifact does not embody a certain aspect of DSD. Italics indicate new elements in DSD (n) as compared to DSD (n-1).

Figure 3. Complete DSD representation of the PaTerm interface. The documentation that goes with DSD (2) is: Figure 1 of DSD (2) is Fig. 1 in this paper. Figure 4 of DSD (2) is Fig. 4 in this paper. Table 1 of DSD (2) is Table 1 in this paper.

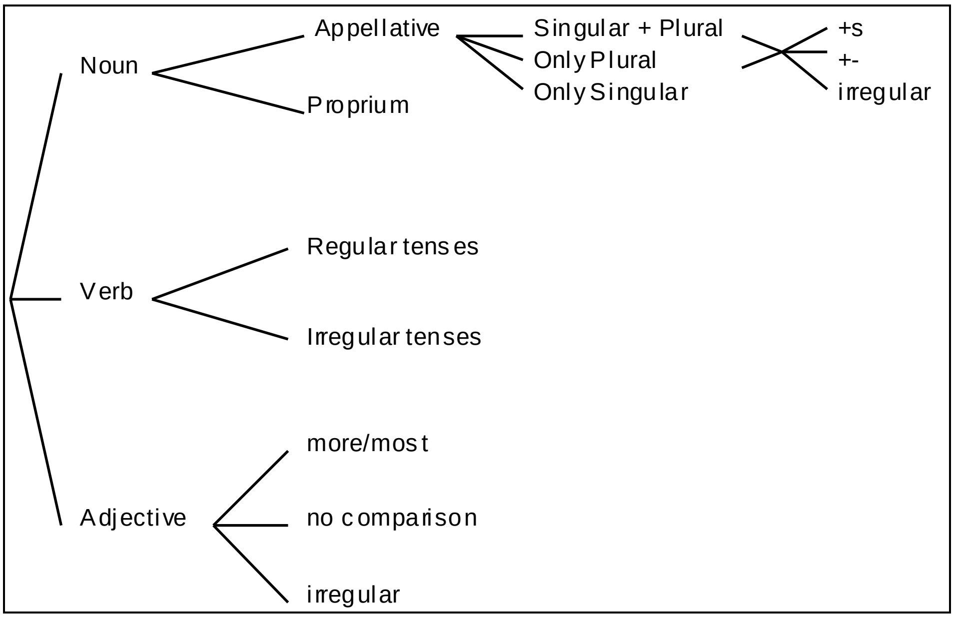


Figure 4. Part of the finite state graph used by the PaTerm system to support the entry of grammatical data.

5. Results of the Information Mapping Step and the Trade-offs

Steps 4 and 5 of IMAP are closely related and their completion may be treated as an integrated process just like Steps 1 to 3. The information mapping and trade-off process, however, is not iterative, but cumulative. A decision made in a trade-off at an early stage may influence subsequent steps in the information mapping process. We will do the information mapping by considering each of the user tasks separately, but let us first make some remarks about the general layout of the tool.

5.1 General layout.

The general layout provides the background - or substrate, in the terminology of Hovy and Arens (1990) - for the information exchange tasks that we will be discussing. The designers' decision to use OSF-Motiv™, which is an X-Windows package, imposes the use of a WINDOW for displaying the PaTerm tool. Reasons for using windows may be high recognisability, as windows have become a de-facto standard in GUI-design practice, and ease of multi-tasking, as PaTerm can be used in parallel with PaTrans. The window should allow access to the main tasks as described in Step 1 of the IMAP analysis: the window should include access to the HELP function which is to be continuously available, and to the main MENU and other MENU's such as dictionary search, dictionary maintenance and management. The window should also display the TITLE PaTerm and the title of the application that one is currently in. It should furthermore allow users to EXIT and to choose a USER LEVEL. All these features can be represented in a number of ways, using different EXPLICIT

STRUCTURES as functional separators as well as for distinguishing foreground from background. The current PaTerm interface can be seen in Appendix 3 Figs. A and B.

5.2 User Task 1. - Find English lexical item to enter. This task does not involve any information exchange between user and system.

5.3 User Task 2. - Enter English lexical item.

The following IMAP rules apply:

> Represent linguistic information with a restricted semantics for which correct spelling (permanent inspection and correction ability) is important <->
>
> Use static written keyword with a clear font.
>
> Visualise a clear offset of the keyword from the background <->
>
> Use a static graphic explicit structure to surround the entry field.
>
> Allow the entry of a new static written keyword, for which correct spelling is important <->
>
> Use the keyboard to enter the letters of the word.

The background of the task depends on the chosen user level. The background can either be an empty window (novice level) or a form-like structure (expert level) which would contain all the headings of the items to be entered.

5.4 User Task 3. - Display lexical items found during dictionary search.

Rules:

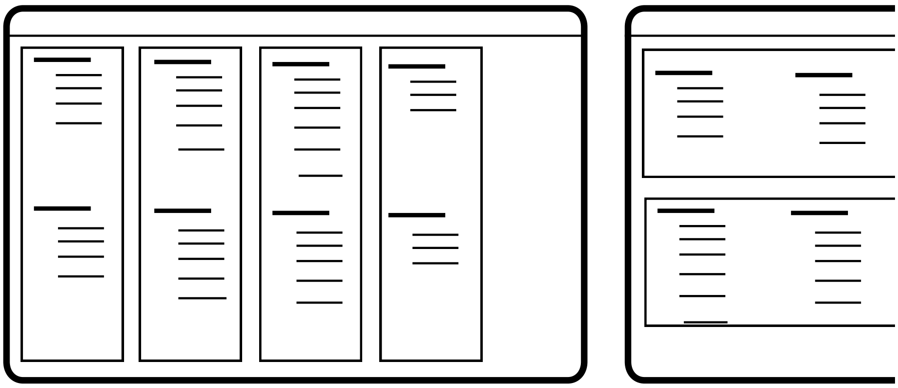
> Represent lists of words and codes, allowing comparison of the lists (permanent inspection) <->
>
> Use static graphic linguistic notation.
>
> Visualise the fact that each list is a separate entity <->
>
> Use a static graphic explicit structure for each list.
>
> Visualise the fact that the lists contain items which are more important than others <->
>
> Use an information channel of static graphic language to express this saliency.

Visualise the importance of the differences between the lists <->

Use an information channel of static graphic language to express this saliency.

Allow comparison of lists <->

Display all lists on one screen.



(a) (

The background is again dependent on the chosen user level, although less so than in the previous user task. The use of form-filling for this task does not allow the lists to be presented on a single screen.

Figure 5. Possible explicit structures for the representation of lists of lexical entries.

Possible explicit structures for representing lists of lexical entries are shown in Fig. 5.

The justification for choosing option (a) in this trade-off is the fact that the uni-dimensional vertical list-structure allows items with a similar meaning to be horizontally aligned, which simplifies comparison across lists.

Fig. 6 shows some possible ways to mark the important items in each list, using different available information channels.

RECORD *noun* +count *-abstract* records ... ***Platte*** +count Platten fem ...	**RECORD** noun +count +abstract records ... ***Rekord*** +count Rekords masc ...	**RECORD** verb reg trans prep in ... ***aufnehmen*** irreg =nahm auf =aufge- nommen trans prep in ...

Figure 6. Some possible ways of marking the differences between lists of lexical items.

The trade-off between these alternatives is not easy. It seems clear, however, that too much marking diversity is confusing and hence counterproductive. The perception of saliency may furthermore be subjectively conditioned. Our personal favourite for this trade-off is option (b).

5.5 User Task 4. - Allow the user to select one of the lists as a template or choose to enter a completely new list.

This task is an example where input and output modalities influence each other thoroughly. The IMAP rules that apply are:

> Allow input of a choice between a limited number of items <->
>
> Represent selectability.

Visualise selectability with keyboard as input device <->

Represent keyability.

Visualise selectability with voice as input device <->

Represent speakability.

Visualise selectability with mouse as input device <->

Represent clickability.

Visualise selectability with touch sensitive input as input device <->

Represent touchability.

Keyability and speakability involve the entry of a recognisable symbol (usually a letter or a digit), which needs to be clearly present in the representation of the items between which the user has to select. Clickability and touchability involve the presence of a sensitive field in the representation of the items (usually some kind of button). Notice that the push-button metaphor relates this representation to 'the real world', and thus enhances recognisability.

The main problem with the representation of this task is the layout of the screen. Some possible screen layouts are presented in Fig. 7.

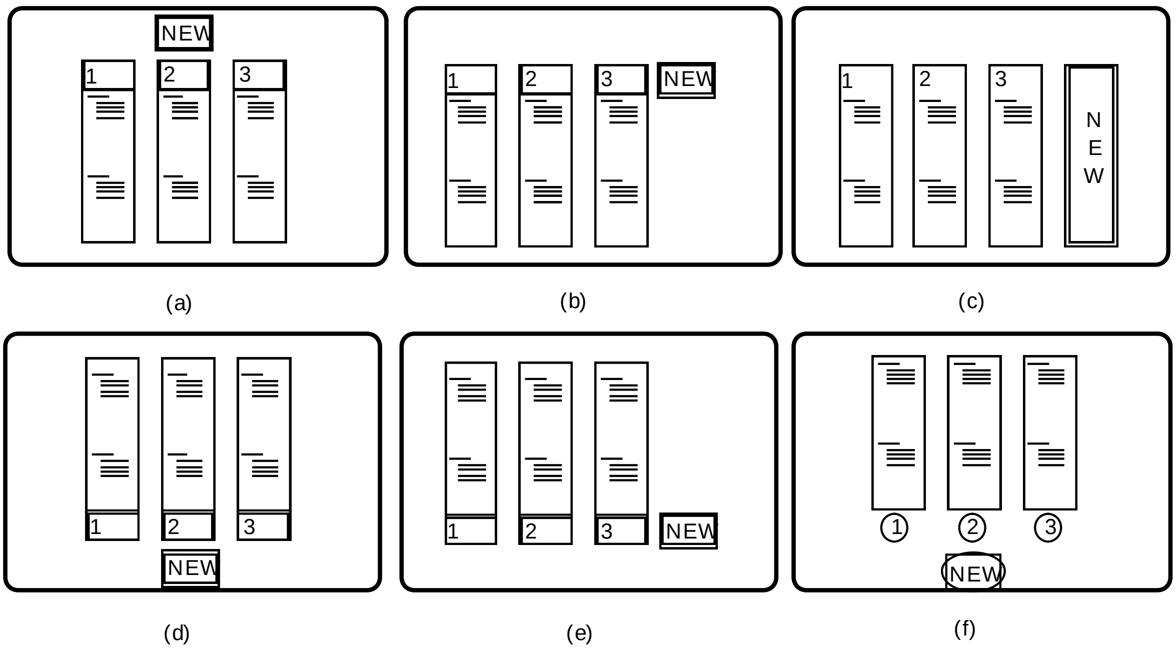


Figure 7. Some possible layouts for User Task 4: Select an item as template for entry. A bold line indicates the presence of a button. A double line indicates the default button. In alternative (c) each list as a whole is an active button.

The trade-off between these alternative layouts is represented in a DR/QOC frame in Fig. 8.

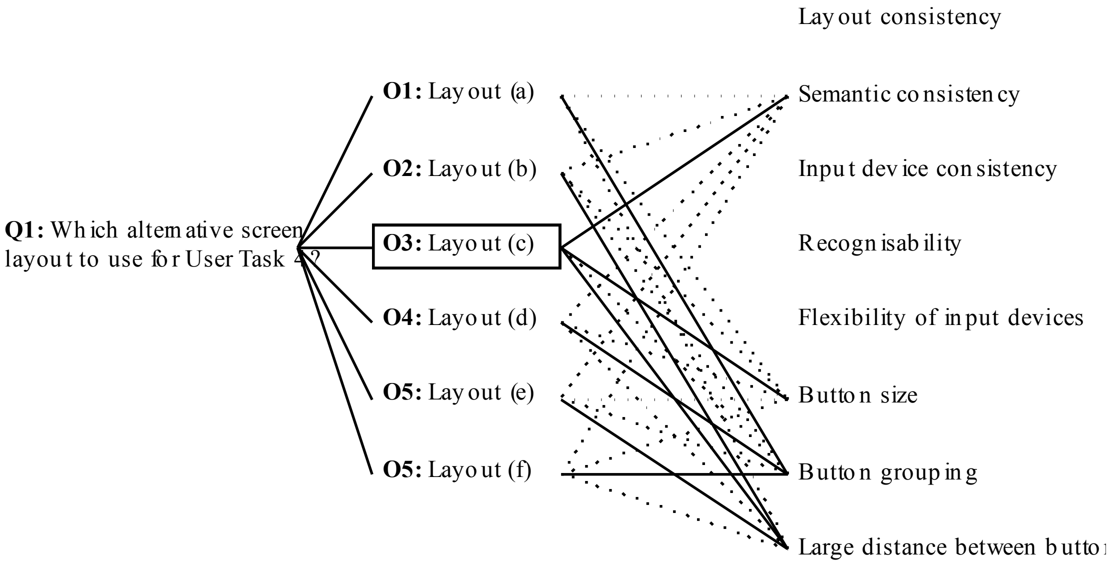


Figure 8. DR/QOC representation of trade-off between design options.

The criteria used in Fig. 8 are the following:

- *layout consistency:* using a similar layout throughout the complete task cycle enhances usability.
- *semantic consistency:* all items representing things with a similar meaning, should 'look' similar.
- *input device consistency:* when one is using a keyboard most of the time, the use of a mouse may hamper the fluency of the task cycle.
- *recognisability:* use de-facto standards when they allow the same functionality and usability as possible alternatives.
- *flexibility: of input devices* allow as many input devices as possible.
- *button size:* large enough buttons are easier to click - and touch - than small buttons.
- *button grouping:* makes the use of the mouse easier.
- *distance between buttons:* a large enough distance between small buttons makes touch screen easier.

Fig. 8 should be interpreted as follows. A full line between an option and a criterion indicates that the option scores positively against the criterion, dotted lines indicate that the option scores neutrally or negatively. The absence of lines between options and criteria indicates the fact that the criterion does not support a motivated choice: the criterion applies to all options equally. The option surrounded by a box is the option that 'wins' the trade-off.

Justification: The criterion of layout consistency applies in so far as all lists are represented vertically, as in previous screen layouts (Figs. 5-7). The criterion of input device consistency would favour keyboard input, but keyboard input - using hotkeys - is possible in all alternatives, so this criterion does not support a motivated choice. In fact, all alternatives offer the same flexibility of input devices. The criterion of recognisability applies equally to all alternatives as each of them uses buttons, which have become familiar to computer users. The importance of the other criteria is clear.

5.6 User Task 5. - Find grammatical information. This task does not involve any information exchange between user and system.

5.7 User Task 6. - Enter grammatical features of the English lexical item.

The use of the finite state net (cf. DSD (2) and Fig. 4) makes the task of entering grammatical features a matter of choosing items from a restricted list of possibilities, except in cases where exceptional grammatical forms have to be entered.

For this task the choice of user level plays an important role. Since the main goal of designing PaTerm was to allow non-linguists to enter items and their grammatical features, the design team decided to offer the beginner's level user examples for each grammatical feature to be entered. The team agreed that at the expert level these examples would be superfluous and probably annoying. In the next paragraphs, we will first discuss the representational aspects at the beginner's level and then those at the expert level.

5.7.1. At the *beginner's level*, the tool should offer the following pieces of information: a description of the grammatical feature to enter, and for each feature option the internal code of the option, a description of the option and a sentence containing an example of the option. The interface should allow the user to choose between options. When necessary, the interface should allow the user to enter other kinds of information, esp. in the case of exceptional grammatical behaviour, such as special plurals for nouns, or irregular flection for verbs. The amount of information that needs to be represented for each feature at the beginner's level, suggests the use of successive screens as one moves along the nodes of the finite state net. The user does, however, need to know what information he/she has entered in previous steps, since this information influences his/her choice for the next feature. This means that immediate feedback, which needs to be permanently presented, is necessary, as is the constant possibility of making changes or backtracking on information.

The following IMAP rules apply:

Represent linguistic information consisting of sentences as well as codes, permitting permanent inspection <->

Use static graphic text/notation.

Visualise semantic clustering of different pieces of information (clear separations between options) <->

Use static graphic explicit structures for each chunk of information.

Fig. 9 shows two possible explicit structures for representing these chunks of information.

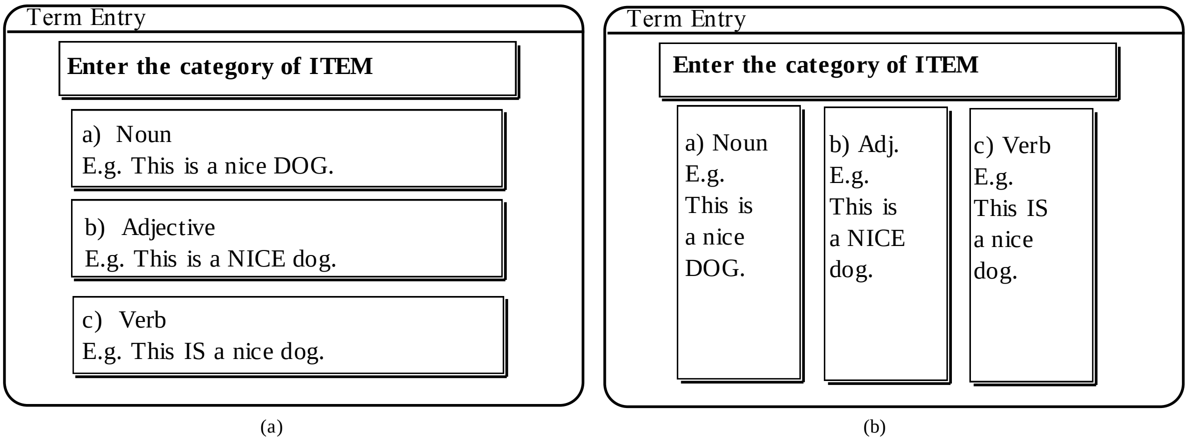


Figure 9. Choosing between vertical and horizontal list representations for the feature options on the beginner's level term entry screen.

The trade-off between these visualisations is presented in Fig. 10.

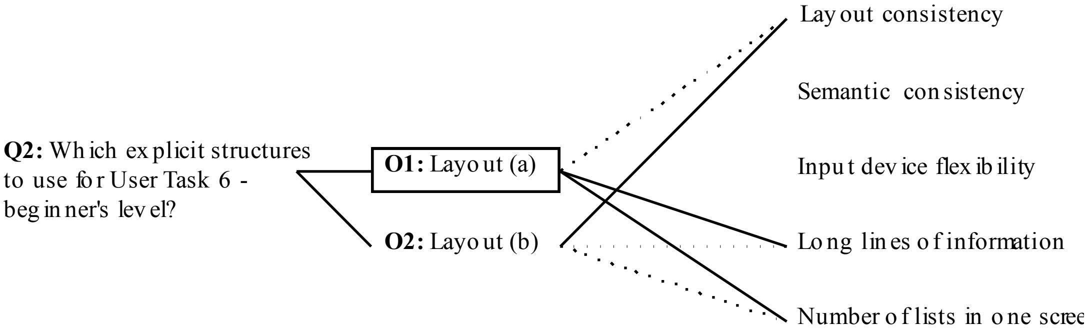


Figure 10. Trade-off between the options of Fig. 9.

Justification: The layout consistency criterion favours vertical lists as used in the previous tasks. The semantic consistency criterion does not favour either option, since the lists in this user task do not represent the same meaning as the lists in User Task 4. The representations in (a) and (b) allow the same input device flexibility as the lists in the previous tasks (keyable, speakable, clickable and touchable). Vertical lists can only represent short lines of information, and the number of vertical lists in one screen is more limited than the number of 'flat' horizontal lists.

The complete layout for this task also includes feedback and the possibility to correct mistakes (cf. above). These two sub-tasks can be supported in a single representation through using a list (as in User Tasks 3-4) in which the elements are active, so that the user can choose to correct a previous choice. The use of the finite-state net will cause each correction of a choice to lead down new paths in the net. The complete layout may then look as in Fig. 11.

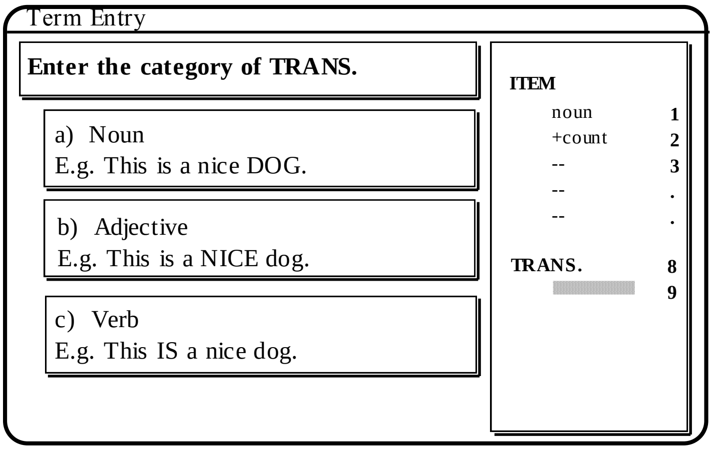


Figure 11. A possible complete layout for User Task 6 - beginner's level.

The choice of the vertical list for feedback/correction in Fig. 11 is motivated by:

a) semantic consistency (the meaning of the list is the same as for User Task 4).

b) layout consistency (the vertical list is familiar to the user).

The numbers in the right-hand side list may serve as hotkeys for the correction (for keyability and speakability), and the features in the lists can be active buttons (for clickability/touchability).

5.7.2. At the expert level, a different representation is needed since all the relevant information for an item should be displayed on one screen. The form-filling metaphor implies that each feature has a fixed position on the screen. The use of a form enhances recognisability: users (in the western world) are used to filling in all kinds of forms. A form can be regarded as a complex static graphic explicit structure. The information to be represented in the form is static graphic notation just as at the beginner's level.

The form-metaphor imposes limitations on how the explicit structure may look. Forms mostly are rectangular pages with fillable slots. The length and number of the slots determine the shape of the complete form. Again, there is a choice between vertical and horizontal structures, but now at different levels. At the highest level, the representation may look as in Fig. 12.

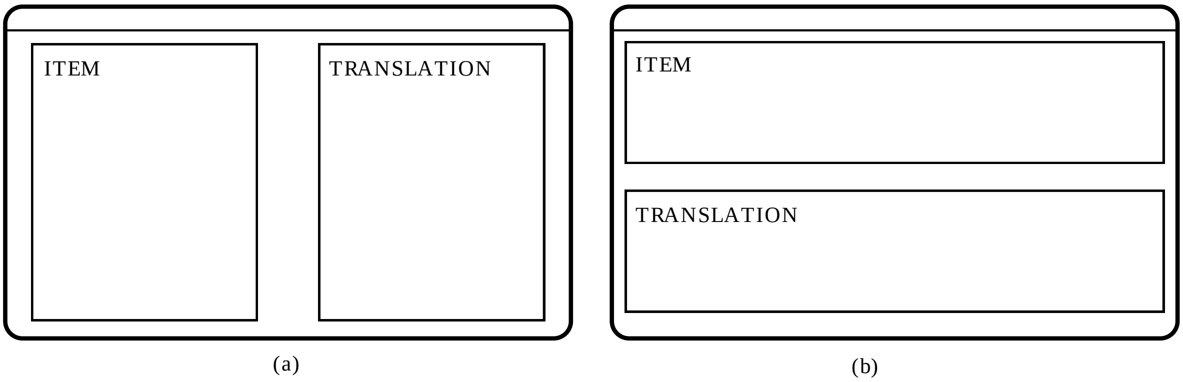


Figure 12. The high-level form structure may be vertical or horizontal.

Whether a motivated choice between these alternatives is possible, cannot be answered at this stage. Possible criteria may be: length of the slots in each main part, importance of hierarchy (vertical alignment seems to introduce more hierarchy than horizontal alignment), layout consistency and semantic consistency. The clustering of semantically related information calls for an explicit structure around each feature bundle, containing the feature description and the chosen option. Fig. 13 shows some possible feature bundle structures.

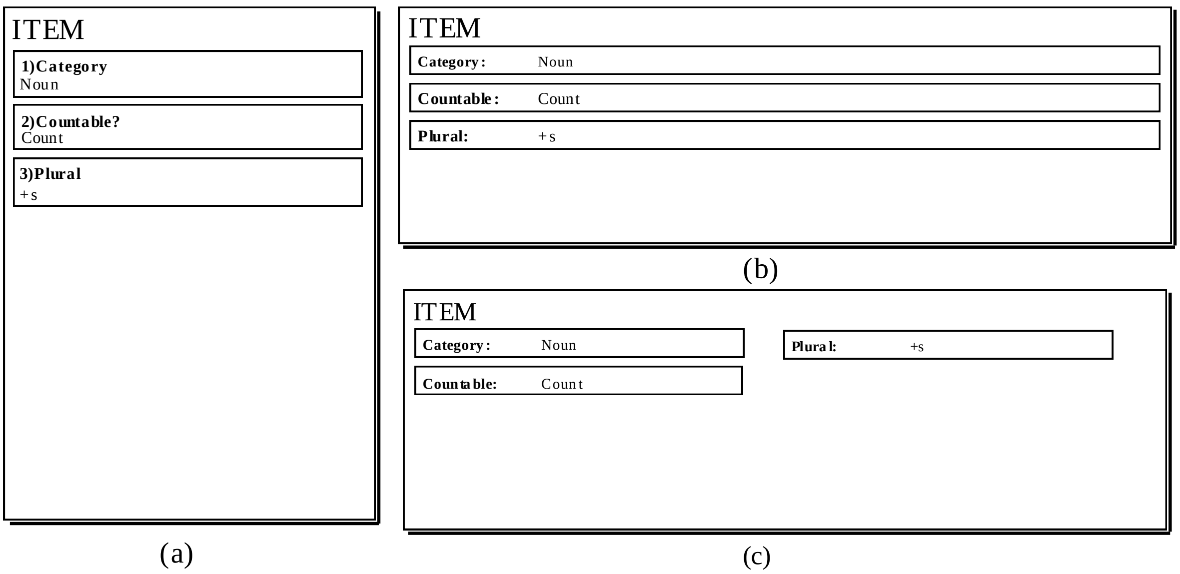


Figure 13. Semantic grouping within a large structure. Choice between (a), (b) and (c) depends mainly on the length of the text that has to fit into the frames.

The choice between options (a), (b) and (c) is represented in Fig. 14.

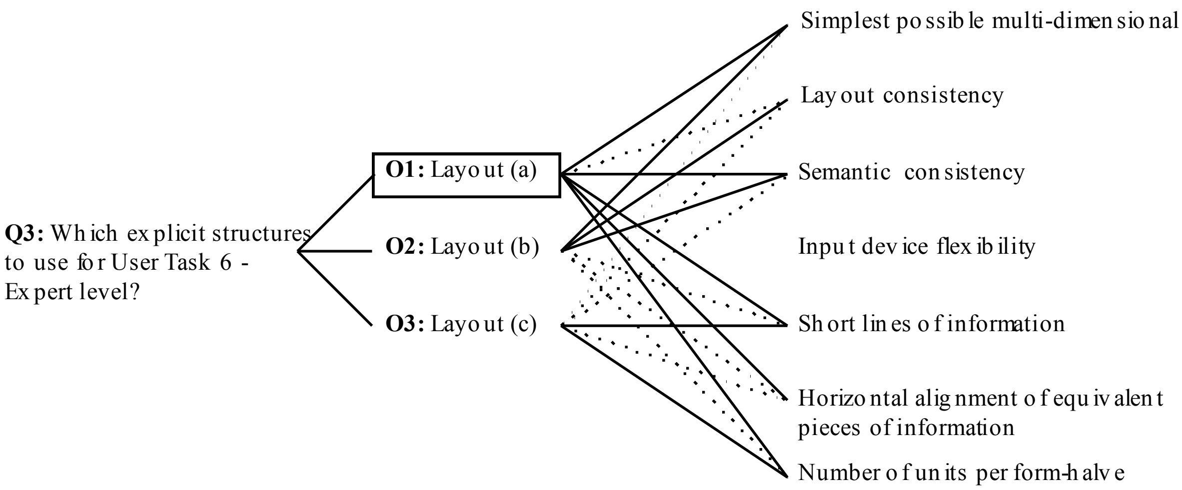


Figure 14. Trade-off of User Task 6 - Expert Level layout, cf. Fig. 13.

Justification: Item (c) has the most complex multi-dimensionality: overall horizontal structure + vertical and horizontal arrangement of items. Item (b) scores best for layout consistency: horizontal organisation of items. Semantic consistency is hard to determine for this trade-off: grammatical features are represented in horizontally organised lists. Flexibility is guaranteed for all three options. Since the grammatical information at this user level consists of codes only, the slots do not have to be very long, so this rules out option (b). Option (a) furthermore allows horizontal alignment of equivalent information in ITEM and TRANSLATION. Finally, options (a) and (c) allow more pieces of information per form halve.

An important feature when using a form, is the use of the available information channels. When there is so much information in one screen, one needs to highlight the more important information This will be discussed below (User Task 11).

5.8 User Task 7. - Find Trans(item). This task does not involve any information exchange between user and system.

5.9 User Task 8. - Enter Trans(item).

There are no special problems of representation, cf. User Task 2.

5.10 User Task 9. - Find grammatical features for Trans(item). This task does not involve any information exchange between user and system.

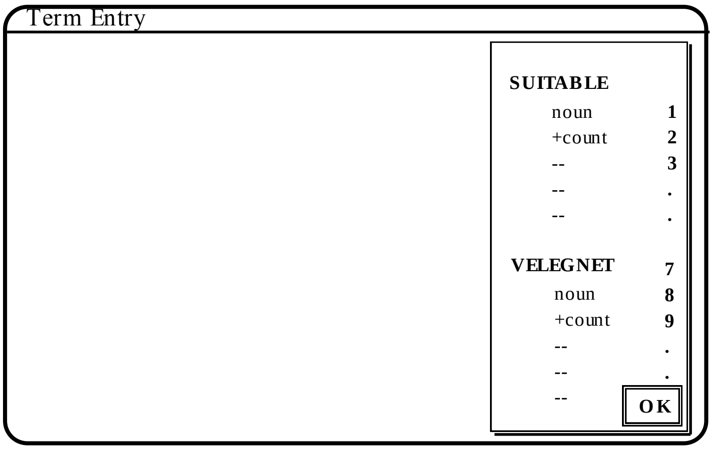


Figure 15. Possible screen layout for User Task 11.

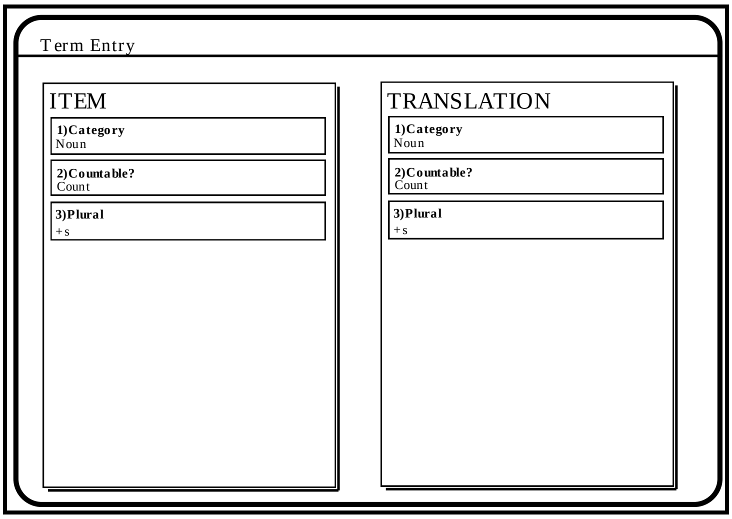


Figure 16. The present PaTerm solution for User Task 11.

5.11 User Task 10. - Enter grammatical features for Trans(item).

There are no special problems of representation, cf. User Task 6.

5.12 User Task 11. - Check result of tasks 1-10.

Again, distinction must be made between the beginner's level and the expert level.

At the beginner's level this task can be performed by keeping the same screen layout as in the previous tasks. This would look as in Fig. 15.

All consistency criteria have been obeyed in this representation.

At the expert level the information channel problem appears. Fig. 16 shows an approximation of the PaTerm designers' solution. The screen contains a large amount of information, and the important information (the options that were entered for each grammatical feature) is not very salient.

An alternative marking strategy is represented in Fig. 17.

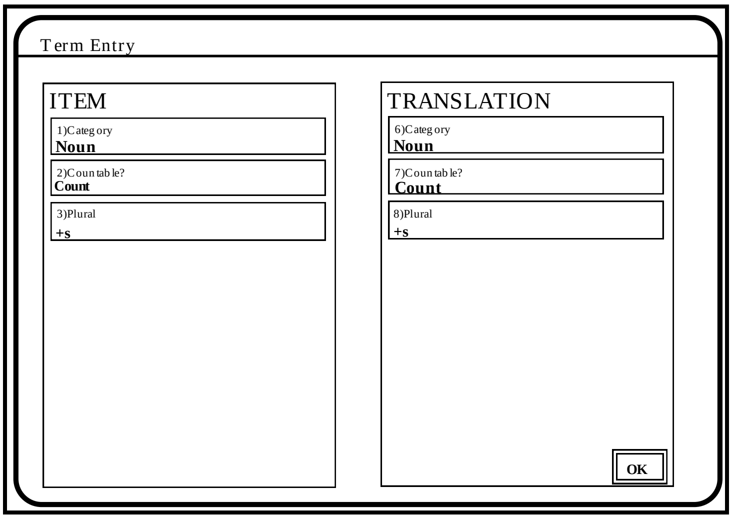


Figure 17. An alternative marking strategy for User Task 11 in which the relevant information is highlighted and the rest is 'grayed-out'.

Once the user has checked the information present in either of the layouts, he/she can either press the OK button or correct some of the information as in the previous user tasks.

5.13 User Task 12. - Compare with existing items.

The user may choose to compare the new item to the already existing ones (if any) before finally adding the item to the dictionary. This user task is similar to User Tasks 3 and 4. The only addition needed is a SAVE button/label (User Task 13). The system will check the database for identical items before saving the new item.

6. Concluding Discussion

6.1 Observations on the information mapping methodology

The information mapping study reported here is the first comprehensive study in which IMAP has been applied to a full-scale realistic design process. A previous study (Bernsen and Bertels 1993) addressed (i) a few high-level problems in the design of a spoken language dialogue system and (ii) detailed problems in a 'toy world' design process of a 'water bath system'. The current study therefore provides a first opportunity to glimpse how IMAP might be applied in real design practice. However, as has been remarked already, IMAP was applied to PaTerm after its completion. This is not a realistic use of the methodology as we were forced into a complex reconstructive exercise which has no correspondence in realistic design. There, IMAP may be expected to be used in parallel with other efforts in artifact specification and refinement. The following comments offer some preliminary observations on the use of IMAP for PaTerm information mapping.

In the present study, IMAP has been applied together with the DSD design space representation and a modified version of the DR/QOC design rationale representation. This raises the question whether IMAP should be developed in such a way as to *require* the parallel use of IMAP, DSD and DR which are all next generation design support approaches. DSD is presently undergoing rapid development, including the incorporation of a design rationale representation which is significantly different from DR/QOC (Bernsen and Ramsay 1994a,b,c, Ramsay and Bernsen 1994).

By itself, IMAP requires a task and task domain analysis which is not importantly different from standard requirements capture. In addition, however, IMAP requires an *explicit* representation of the results of the task and task domain analysis to enable a controlled application of Modality Theory during the information mapping phase. We believe that DSD is a suitable candidate for this purpose. Like a previous IMAP study (Bernsen and Bertels 1993), the present study demonstrates how extensive and how varied is the information which carries potential implications on how the information mapping should be done. This suggests that a 'full' application of DSD to the design process might be what best serves as a basis for IMAP. In other words, the hypothesis is that DSD does not have to be specifically tailored in order to support IMAP. We will explore this hypothesis in future case studies.

In the present study, DR/QOC has been useful in exploring the design space around specific interface design options. Future case studies will investigate whether the DSD design rationale representation is able to do a similar job.

The application of IMAP to the PaTerm interface has been quite detailed. For most sub-tasks of the selected representative task, a detailed information mapping exercise was carried out and explicitly described, including its design rationale representation. A key issue to be addressed in future work is the level of generality or abstraction at which Modality Theory will be able to support interface design (Bernsen, Lu and May 1994). We do not expect Modality Theory to generally deliver rules of a nature as detailed as most of the information mapping rules used in reasoning about the PaTerm interface. This does not mean that reasoning at this level of detail is unimportant in interface design. On the contrary, such reasoning is necessary and is done every day by interface designers all over the world. In other words, the present study points to a possible distinction between the application to interface design of Modality Theory as a science-based design support tool, on the one hand, and the usefulness of making explicit interface design reasoning at a level of detail beyond the scope of Modality Theory on the other. This is no contradiction. Both efforts may be eminently useful. But they are clearly distinct and future work will have to determine their respective scopes of achievement.

The preliminary picture which emerges from the above observations, is the following. DSD, including selective use of explicit design rationale representations for hard interface design problems and trade-offs, seems to work as an integral part of IMAP. IMAP requires a full DSD representation rather than an application-specific, ‘truncated’ version of DSD. But whereas DSD can be used throughout, to any level of interface design detail, Modality Theory cannot be expected to support all of the problem solving handled by DSD/IMAP. There is nothing really surprising about this tentative conclusion, as the days are long gone when HCI expected complete coverage from supporting basic theory.

As to providing input to Modality Theory development, the PaTerm study has been very helpful in pointing towards ways to represent key aspects of input modalities. This will be taken into account in our planned efforts to extend Modality Theory to cover input modalities.

Finally, the present case study clearly shows the iterative nature of IMAP. Iterativity not only appears within the larger units (Steps 1-3 and Steps 4-5), but also between these units. During information mapping we sometimes found that we had to go back to the task analysis steps to gather more information.

6.2 The PaTerm tool - proposals for additions

From the reaction of the PaTerm design team members during the interviews, we gathered that some of the problems raised by our use of IMAP had not been considered by them. Had this been a real design process, the questionnaire that was used as part of the detailed task analysis would have been very useful to the design team. The detailed task analysis lead us to distinguish between three situations in which PaTerm will be used. The study of these situations generated the idea to represent the already available items (cf. User Tasks 3 and 4) in lists on a single screen to allow comparison between them.

A major suggestion for a possible redesign of the PaTerm tool is the addition of User Tasks 3 and 4, i.e. the display of all homographs that are present in the dictionaries. This is particularly necessary in the post-editing situation in which very fine linguistic nuances have to be taken into account. At the moment, PaTerm does not offer the ability to compare several lists on a single screen.

The introduction of the notion of ‘situation of use’ is another important addition. It offers the user some extra flexibility.

Fig. 14 suggests that use of two vertical lists at the expert level (option (a) in Fig. 13) is to be preferred to the use of two horizontal structures (option (c) in Fig. 13). The QOC suggests that (a) is more user-friendly than (b) and (c).

The following remark stems more from intuition than from motivated analysis. The expert representation in the present PaTerm interface does not supply the user with the description of the feature he/she has to enter (cf. the PaTerm screen dump B in Appendix 3). This information seems necessary to clearly show the user what is expected of him/her at a certain stage. For instance, screen dump A (Appendix 3) contains the entries 'Ingen' and 'Endelse' (None and Ending) without any further specification. This may be confusing. A clear task identification seems necessary.

A final remark concerns the use of information channels at the expert level in User Task 11. In the present PaTerm interface, almost all explicit structures and features are highlighted. This makes the screen highly cluttered and potentially confusing. We suggest that all explicit structures and irrelevant pieces of information are 'grayed out' and that only the selected options for each feature bundle are marked, so that the relevant information immediately stands out.

Acknowledgements.

The work reported in this paper was done on a grant from the Esprit Basic Research project AMODEUS II whose support is gratefully acknowledged. We wish to thank the CST PaTerm designers for their kind assistance and their valuable time.

References

Barnard, P.J. (1993): Modelling Users, Systems and Design Spaces (Esprit Basic Research project 3066-AMODEUS). *Proceedings of HCI 93*, Orlando, Florida.

Barnard, P. (1991): Bridging between basic theories and the artifacts of human-computer interaction. In J.M. Carroll (Ed.): *Designing Interaction. Psychology at the Human-Computer Interface.* Cambridge: Cambridge University Press.

Bernsen, N.O. (1993a): A Research Agenda for Modality Theory. In Cox, R., Petre, M., Brna, P. and Lee, J. (Eds.): *Proceedings of the Workshop on Graphical Representations, Reasoning and Communication.* World Conference on Artificial Intelligence in Education, Edinburgh, August , 43-46.

Bernsen, N.O. (1993b): The Structure of the Design Space. In Byerley, P.F., Barnard, P.J. and May, J. (Eds.): *Computers, Communication and Usability: Design Issues, Research and Methods for Integrated Services.* Amsterdam, North-Holland, 221-244.

Bernsen, N.O. (1993c): Structuring Design Spaces. *InterCHI '93 Adjunct Proceedings,* 211-212.

Bernsen, N.O. (1994): Foundations of Multimodal Representations: A Taxonomy of Pure Modalities. To appear in *Interacting with Computers*.

Bernsen, N.O. and Bertels, A. (1993): A Methodology for Mapping Information from Task Domains to Interactive Modalities. *Esprit Basic Research Project GRACE Deliverable* 10.1.3.

Bernsen, N.O., Lu, S. and May, M. (1994): Towards a design support tool for multimodal interface design. The taxonomy workbench and theory demonstrator. *Esprit Basic Research project AMODEUS II Working Paper* RP5-TM-WP5, 1994.

Bernsen, N.O. and Ramsay, J. (1994a): AMODEUS II shared modelling Excercise: DSD modelling technique report on the EuroCODE exemplar. *Esprit Basic Research project AMODEUS II Internal Report* RP3-ID-IR8.

Bernsen, N.O. and Ramsay, J. (1994b): Design structure, process and reasoning. The advancement of a tool for the development of design spaces. *Esprit Basic Research project AMODEUS II Working Paper* RP3-ID-WP?.

Bernsen, N.O. and Ramsay, J. (1994c): A Manual for the Design Space Development (DSD) Framework. *Esprit Basic Research project AMODEUS Working Paper* (in preparation).

Hovy, E. and Arens, Y. (1990): When is a picture worth a thousand words? Allocation of modalities in multimedia communication. Paper presented at the *AAAI Symposium on Human-Computer Interfaces*, Stanford.

MacLean, A., Young, R., Belotti, V. and Moran, T. (1991): Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction,* Vol. 6 (3&4). Special Issue on Design Rationale, Carroll, J.M. and Moran, T.P. (Eds.), 201-250.

MacLean, A., Belotti, V. and Shum, S. (1993): Developing the Design Space. In Byerley, P.F., Barnard, P.J. and May, J. (Eds.): *Computers, Communication and Usability: Design Issues, Research and Methods for Integrated Services.* Amsterdam, North-Holland, 197-220.

Ramsay and Bernsen (1994): Ramsay, J. and Bernsen, N.O.: AMODEUS II Shared Modelling Exercise. DSD Modelling Technique Report on the CERD Exemplar. *Esprit Basic Research project AMODEUS II Internal Report* RP3-ID-IR9.

Appendix 1. Questionnaire

Most of the questions of this questionnaire were answered in the third interview with the designers of PaTerm. The parts of text marked in boldface are the choices made by the designers and the translation company. Notice that not all questions could be answered by the designers.

Pre-task organisation

Problem 1: *PTO - ST* Does the translating firm receive all translatable texts at once, or batch by batch?	(1) all translatable texts at once (2) batch by batch	The best design option seems to be to start from option 2. Td-Db may need later revising. *It seems to be the option chosen by the designers.*

This question seems irrelevant to the design of the tool.

Problem 2: *PTO* How will the texts in $BATCH_n$ be searched for lexical items? How will LIST, the list of all items that will eventually be entered in Td-Db, be drafted?	(1) an automatic search system, comparing words in texts with words that are already in Gen-Db and Td-Db. (2) manually scan texts, looking for new items (3) human-controlled automatic process	- Option 1 will not suffice for different reasons (problems 3-5) - Option 2 seems to be impossible from a practical view. - Option 3 seems to be the only plausible one, for reasons of efficiency and completeness.

Option (1) has been chosen for the following reasons:

- Texts that are being translated, are sent through a document handler before the actual translation. This document handler scans the documents for words and looks for these words in the dictionnaries. A frequency list is made up of all words in the text. In this list the unfound words are put first, so that the person handling the translation system can decide whether or not the unfound words are ITEMS that have to be entered using PaTerm.

- The translated texts are post-edited by an official translator, who will then have to find mistakes due to 'multiple word items, specific use of general terms.

Problem 3: *PTO* The texts in $BATCH_n$ may contain complex lexical items, such as LUBRICATING OIL. How will they be detected?	Assuming that option 3 was chosen for Problem2, they can be detected by an expert in the Client Task Domain. With option 1 for problem 2, They will probably not be detected.	

As described in Problem 2, the detection of multiple word terms is up to the post-editor of the translated text., as is Problem 4. (cfr. infra)

The post-editor will then have to make a new list of items that have to be added to the Td-Db.

Problem 4: *PTO* Lexical items appearing in Gen-Db, but used in a domain-specific way, how will they be detected?	Assuming option 3 for problem 2, they can be detected. Assuming option 1, they probably can not.	
Problem 4b: *PTO* What will happen when the lexical items from Problem 4 are detected?	(1) the items are entered in LIST (post hoc) (2) they are not entered in LIST	Option 1 seems to be the best solution, since translations will probably be better when all domain-specific items can be translated properly.
Problem 4c: *PTO - S* What is the hierarchy of the databases? Will Td-Db be searched first? From specific to general?	(1) Td-Db is searched first. If ITEM is found, Gen-Db is not searched. (2) Gen-Db is searched first.	Option 1 seems to be the only reasonable solution for this translation system.

Problem 5: *PTO - S - UT* The task domain may contain items that are general enough to be added to Gen-Db. Are changes to Gen-Db allowed?	(1) Gen-Db cannot be changed (2) Gen-Db can be changed by the ADMINISTRATOR (3) Items can be added to Gen-Db by any users.	Solution 2 seems to be the optimal one, if changes are allowed. But who decides when an item is general enough? If Gen-Db is updated, what happens to other copies of Gen-Db? A general upgrade? - Maybe the best solution is NOT to change Gen-Db after all. (For practical reasons.)

When items such as those described in Problem 5 are found, no one in the translating firm, can enter them in Gen-Db. The latter dictionnary can ONLY be altered by the designers of PaTrans and PaTerm, namely CST. If such items are found, that has to be reported to CST, who can then make the necessary changes.

User Tasks

The user gets a list of English lexical items (LIST) that have to be added to a non-empty Td-Db.

As described at the beginning of this paper, this task description is in fact not very accurate, although the algorithm described below seems to fit the actual task anyway.

For item$_n$ in LIST (0 < n < 701), do

1) *Enter item$_n$ in Td-Db.*

2) *Find the necessary grammatical information for item$_n$.*

3) *Enter this info in Td-Db.*

4) *Find trans(item$_n$).*

5) *Enter trans(item$_n$) in Td-Db.*

6) *Find the necessary grammatical information for trans(item$_n$).*

7) *Enter this info in Td-Db.*

8) *Check result of steps 1) - 7).*

9) *Save all the information for item$_n$.*

1) Enter item$_n$ in Td-Db.

Problem 6: *U*		
Who is the user?	(1) a linguist with expertise in English. (2) a linguist with expertise in Danish (3) both 1 and 2 (4) an expert in the CLIENT task domain (5) both 1 and 4 (6) both 2 and 4 (7) both 3 and 4	For some Problems listed below, a substantial amount of linguistic background in both English and Danish is required. (in translation practice, a rule of thumb is that one should only translate to one's own language, so the user is preferably a Danish native speaker). For other Problems, a substantial amount of Task Domain knowledge is needed. Basic knowledge of the Task Domain seems to be mandatory.

The user of PaTerm is not supposed to be a linguist. What is needed is the following:

- preferably native Danish (since the interface is to be in Danish and one is translating into Danish).

- some linguistic background is necessary, but high-school level should suffice to use the tool. All possible problems will have to be explained and exemplified.

- the user should know the task domain he/she is working in, but for very specific terms, counselling may be advisory.

- the user should have a good knowledge of English.

The profile of the persons who will be using the system in practice, did not seem to be known at the design phase. ***????***

Problem 7: *UT - U* ITEM can have multiple interpretations. Which of these interpretations will be entered?	(1) only the interpretation which appears in $BATCH_n$ (2) all possible interpretations of ITEM (3) only those interpretations which are relevant to the task domain	criteria: - completeness - relevance - efficiency = search time

This problem should not occur, since what is entered using PaTerm, is a term in the narrow, unambiguous sense of the word.

However, if there are multiple interpretations, all relevant ones should be entered. (by CST ***???****)*

Problem 8: *O - TC* Does the user have access to $BATCH_n$? Is that necessary?	(1) No (2) Yes, in printed form (3) Yes, in electronic form	$BATCH_n$ may provide background as to the usage of certain lexical items. So 2 or 3 seem useful.

Problem 9: *TC - S* Does the user have access to background information, lexical and/or grammatical support, such as dictionaries, grammars, ...?	(1) No (2) Yes, in printed form (3) Yes, in electronic form (4) Yes, in the form of phonenumbers of professional translators.	Some form of support is expected to be needed at some time or other. Maybe a combination of 2, 3 and 4 would be best.

Provided that such specialized background information exists, it will be available to the user. If no printed material exists, counselling may be called upon. (cfr. Problem 6)

Problem 10: *TC* Do such very specialised grammars and/or dictionaries exist?	(1) No (2) Yes	One can only hope for option 2 to be the case.

The interviewees answered No to this question. The number of good specialised Danish dictionnaries seems to be limited.

Problem 11: *UT* Does the user check every lexical item for multiple interpretations?	(1) Yes (2) No, only the ones of which he knows that multiple interpretations exist. (3) No. If BATCH contains multiple entries of an item, then all interpretations in BATCH are entered.	criteria: - efficiency (look-up can be quite time-consuming) - user-expertise in Task Domain - For option 3, some way of searching Td-Db must be available (before entering new interpretations?)

TERMS can not have multiple interpretations.

Problem 12: *UT - U* Which criteria does one use to decide whether or not a certain interpretation is relevant to the task domain?	(1) Personal background knowledge (2) Counselling by task domain experts (3) Frequency of the item?	This is a tough one ...

Problem 13: *I* Which is the best way to enter ITEM into Td-Db?	(1) Speech (2) Typing (3) Choosing from lists ...	This problem is to be directed in detail later...

Problem 14: *UT* Does one enter the AE or the BE spelling?	(1) Am. Engl. (2) Br. Engl. (3) The spelling used in $BATCH_n$	Again a tough problem to tackle...

There was no clear answer to this question. One would use the British English spelling as default and standard, but what would happen when BATCH contains American English items has not been thought about.

Problem 15: *S - UT* Is there some form of pre-checking when ITEM has been entered?	(1) No (2) Yes	From the discussion of the previous problems, one would be inclined to opt for (2).
Problem 15b: *I* Assuming option (2) in the previous problem, how can one represent ITEMS that were already present?	(1) As a list containing only items (no further info) (2) As a list of items with their full information (3) Lists, one by one (4) simultaneous lists	Again, this problem will be thoroughly analysed later.

When asked this question, both interviewees agreed that such pre-checking would probably be an interesting feature. It didn't seem to have crossed their minds before.

2) Find the necessary grammatical information for item$_{n}$.

Problem 16: *UT - S* Which grammatical information is entered?	(1) all possible grammatical information (2) only the information that will be used by the PaTrans.	Option (2) seems to be the obvious answer to this question for reasons of efficiency and consistency.

PaTrans is based on the EUROTRA translation model, since it was the most extensive model available, and was very robust during application.

3) Enter this (grammatical) info in Td-Db.

Problem 17: *UT - S* How does the user enter this grammatical information?	(1) in random order (2) guided by the system	Option 2 seems to be the best solution for reasons of consistency and completeness.

Problem 18: *I* What does the entry screen look like? (= output modalities)	(1) plain white screen (2) window containing cells which need to be filled in. (3) window showing lists of possible answers	Needs thorough analysis.
Problem 18b: *I* **What input modalities are used?**	(1) keyboard (2) mouse (3) touchscreen (4) a combination of 1, 2 and 3	criteria: - availability (machine restrictions) - feasability (implemention time constraints) - efficiency (= speed?) - user expertise - user friendliness

Touchscreen was described by one of the interviewees as messy. One would need gloves when using touchscreen to avoid the screen from getting very dirty and unreadable. Second, the interviewee found it hard to point to a screen (lack of context, i.e. he will first put is hand on the edge of the screen - for reasons of steadiness and coordination - and only then touch the screen.) The interviewee did not like the idea of touchscreen.

Problem 19: *S* Does the user get feedback?	(1) Immediate feedback (2) Feedback when all information is entered (3) No feedback	Option 1 seems to be preferable.

Problem 19b: *S* Can he/she correct mistakes?	(1) immediate correction (2) correction when all information is entered (3) both 1 and 2 (4) no correction: deletion and restart	Option 3 is probably preferable for reasons of user-friendliness and efficiency.

Both feedback and correction are available at all times during and after coding. This greatly enhances flexibility and user-friendliness. The user feels in control at all times during coding.

4) Find trans(item$_n$).

Problem 20: *UT* Is there only one possible translation?	(1) Yes (2) No (=> cfr. Problem 7: Enter all possible translations?)	In the case of (1), there is no problem. In the case of (2), there is.
Problem 20b: *UT - TC* Is there any documentation available to help the user decide which translation to code?	(1) No (2) Yes, in printed form (3) Yes, in electronic form	Again, let's hope that either 2 or 3 is the case.

- In the narrow sense of the term TERM, there should only be one translation. If, however, there are several translations, they have to be entered as separate lexical items. The translation system will then syntactically disambiguate them, and present all remaining translations to the post-editor.

- There are some normal dictionnaries, but the amount of specialised English-Danish dictionnaries seems to be limited.

Problem 21: *UT* What happens if no translation exists?	(1) ITEM is not entered in Td-Db (2) The nearest translation is entered.	Tough problem...

The EUROTRA-system has been adapted to handle untranslatable words. On top of Eurotra, a fail-soft mechanism pre-edits the text (a specific pre-editing tool is still under development). In its normal implementation, the Eurotra-system would crash if it met with an untranslatable item. Thanks to the fail-soft mechanism, the item is put between quotes, and the whole sentence is translated only in a preliminary way. It is up to the post-editor to check the translation.

Problem 22: *UT* The English ITEM is not translated, simply copied in Danish. (as many technical terms are)	(1) ITEM is not entered in Td-Db (2) ITEM is entered in Td-Db, together with its Danish usage.	

Problem 22b: *UT - S* The English item can: 1- take over Danish inflection and usage 2- take over Danish usage, but retain English inflection 3- retain English usage and inflection	(1) the system can handle all these different cases (2) the system can only handle the 'normal' case (3) the system allows for some irregularities, but will not handle all.	Possible decision criteria: - feasibility - frequency of boroughed items - is there any post-editing to be done on translations?

All the different stages in lexical borrowings can be handled in the system.

5) Enter trans(item$_n$) in Td-Db.

6) Find the necessary grammatical information for trans(item$_n$).

7) Enter this info in Td-Db.

Steps 5) - 7) in the 'algorithm' do not seem to pose any new problems.

8) Check the information from steps 1) - 7).

Problem 23: *I* How is all this information represented, if it is to be checked quickly?	(1) in a single list (2) on one screen (3) several lists, containing the new item, next to already present homographs. ...	This needs a more specific analysis later.

Seen as the system has two user-levels, the information is represented in two different ways. a) as a simple list (in the beginner's level) b) as a single screen, where all information has a fixed location (for the experienced level) Especially the last representation offers an interesting topic of study, since it is not very good (according to NOB and SV).

Problem 24: *S* What will the filestructure be like? How is all this information internally represented?		Decision criteria: - flexibility - efficiency - PaTrans data representation

There are two database structures in which the information is stored:

- for translation purposes, the database is only indexed on the English item, so as to allow efficient look-up.

- for coding (and searching) purposes, more indexes are installed in the database, so as to allow flexibel searches.

Appendix 2. DSD (1)

Design Project: PaTerm (completed at the time of the analysis) at the Center før Sprokteknologi at København Universitet		
DSD No. 1	**Date: 03/03/1994**	**Sign: SV**
A. General constraints and criteria		
Overall design goal(s)		
An interactive tool for adding lexical items to the Task Domain dictionnaries of the PaTrans patent-text-translation system.		
General feasibility constraints		
Scientific and technological feasibility constraints		
The tool should run in parallel with the PaTrans translation system, in an X-Windows™ environment on UNIX-machines.		
Design process type		
Commercial (PaTrans and PaTerm have been sold to LingTech, a Danish patent-translation company)		
Designer preferences		
Use of the Motif™ design package for X-Windows™		
Realism criteria		
Functionality criteria		
The translated texts are post-edited by an experienced translator.		
Usability criteria		
B. Constraints and criteria applied to the artifact within the design space		
Collaborative aspects		
4 people in the design team: - 3 linguists, who specify the linguistic properties of the information that is to be entered. - 1 technician, who does the actual implementing. Next to these 4 people, most of the researchers working on the PaTrans system were involved, since PaTerm and PaTrans are to be used in parallel		
Organisational aspects		
- A TERM is defined as a lexical item that must be closely related to the task domain, and that only has one translation. Ambiguous terms are not to be entered by the user. - A list of terms is made by a document handler, which searches the general dictionnary (GenD) and the task domain dictionnary (TaskD) and makes a frequency list of the words that appear in the text that has to be translated (TextT). The words that do not appear in either dictionnary, are to be entered by the user. - CST supports the user of PaTerm. All difficulties, ambiguities, changes to GenD are reported back to CST, which then handles them. - The post-editor can add new TERMS to the TaskD dictionnary after editing the translation. - There are several working TaskD's, but only one official TaskD. The working dictionnaries have to be approved by the Administrator. - The user(s) must have access to TextT, and to support material such as specialised dictionnaries and grammars, if they exist.		
System aspects		
- There is a hierarchy in the use of the dictionnaries in PaTrans. TaskD is scanned before GenD is. - Changes to GenD cannot be made by the users. - The existing dictionnaries can be scanned. - PaTrans uses only a limited amount of grammatical information. This info has a fixed order and form. - PaTrans allows multiple entries for a Term. - The Term data are internally represented in two different ways. One has multiple indexing for searching and coding purposes. The other has only one index, for fast translation purposes.		
Interface aspects		

- Keyboard and mouse input. - Graphical and acoustic output.

Task aspects
System tasks: - Give feedback on existing homographs and coded information, - Allow change and correction. - Check information for completenss and consistency. - Provide help and manuals. **User tasks:** 1) Find TERMS to enter in TaskD. 2) Enter Term in TaskD 3) Find grammatical information for TERM 4) Enter this information in TaskD 5) Find trans(TERM) 6) Enter trans(TERM) 7) Find grammatical information for trans(TERM) 8) Enter this info in TaskD 9) Check result of steps 1 to 8 10) Save result of steps 1 to 9
User and user Experience aspects
The user is preferably a Danish native speaker, with a minimal linguistic background (high-school level should suffice), a good knowledge of English and a thorough knowledge of the Task Domain. There are 4 types of users: - normal users: - beginners - experts - administrator - post-editor
C. Hypothetical issues
Do specialised dictionnaries and grammars (English - Danish) exist? The answer seems to be NO. Should counselling be provided? Does one enter AE or BE spelling or perhaps both? Can touchscreen be useful as an input modality? Is TERM too narrowly defined?
D. Documentation
Linguistic manual Technical manual Simple translation dictionnaries.
E. Key: DSD No. (n) indicates the number of the current DSD frame. 'Null' means that the artifact does not embody a certain aspect of DSD. Italics indicate new elements in DSD (n) as compared to DSD (n-1).

Appendix 3. The PaTerm Interface.

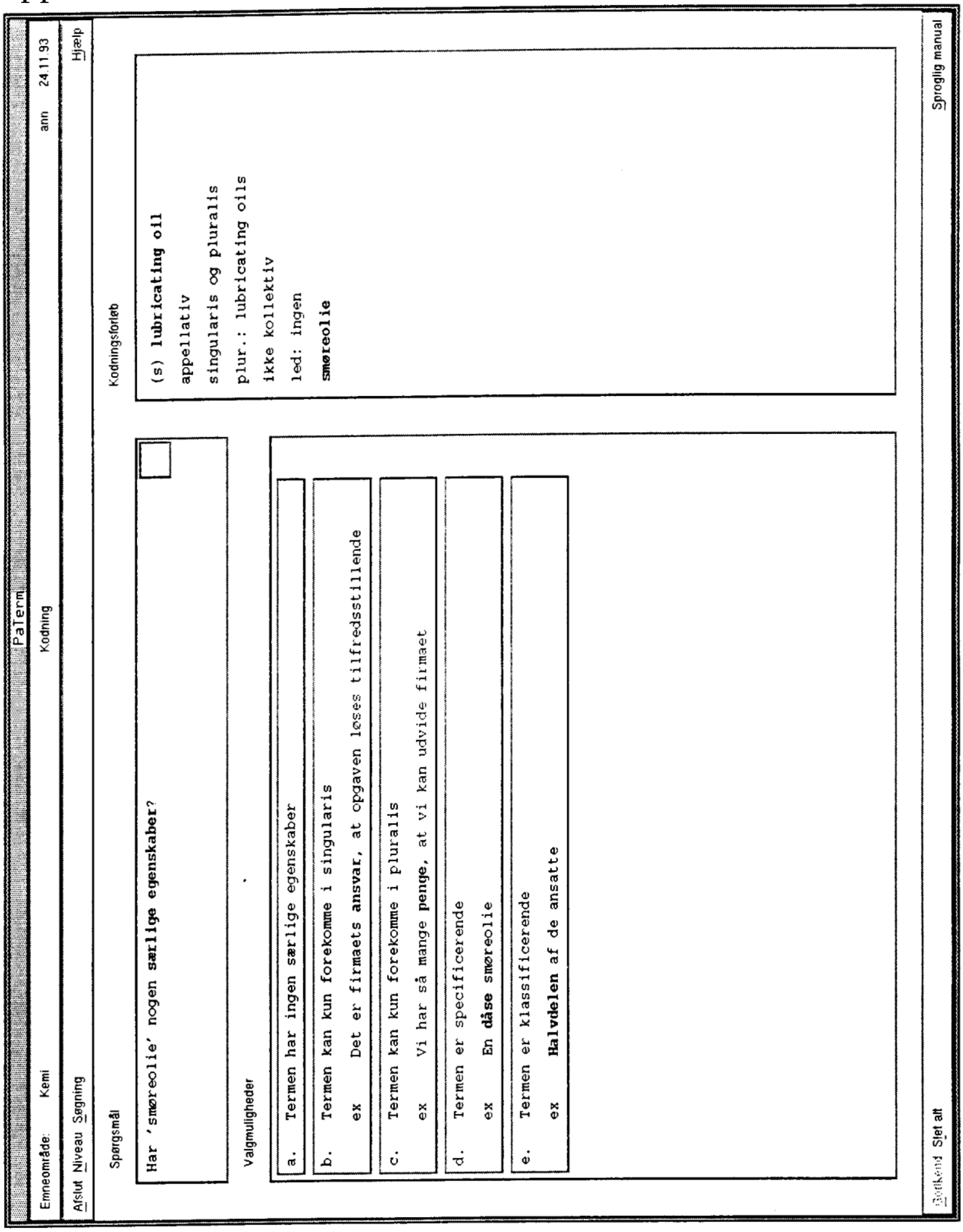


Figure A. Screendump: PaTerm - Term Coding - beginner's level.

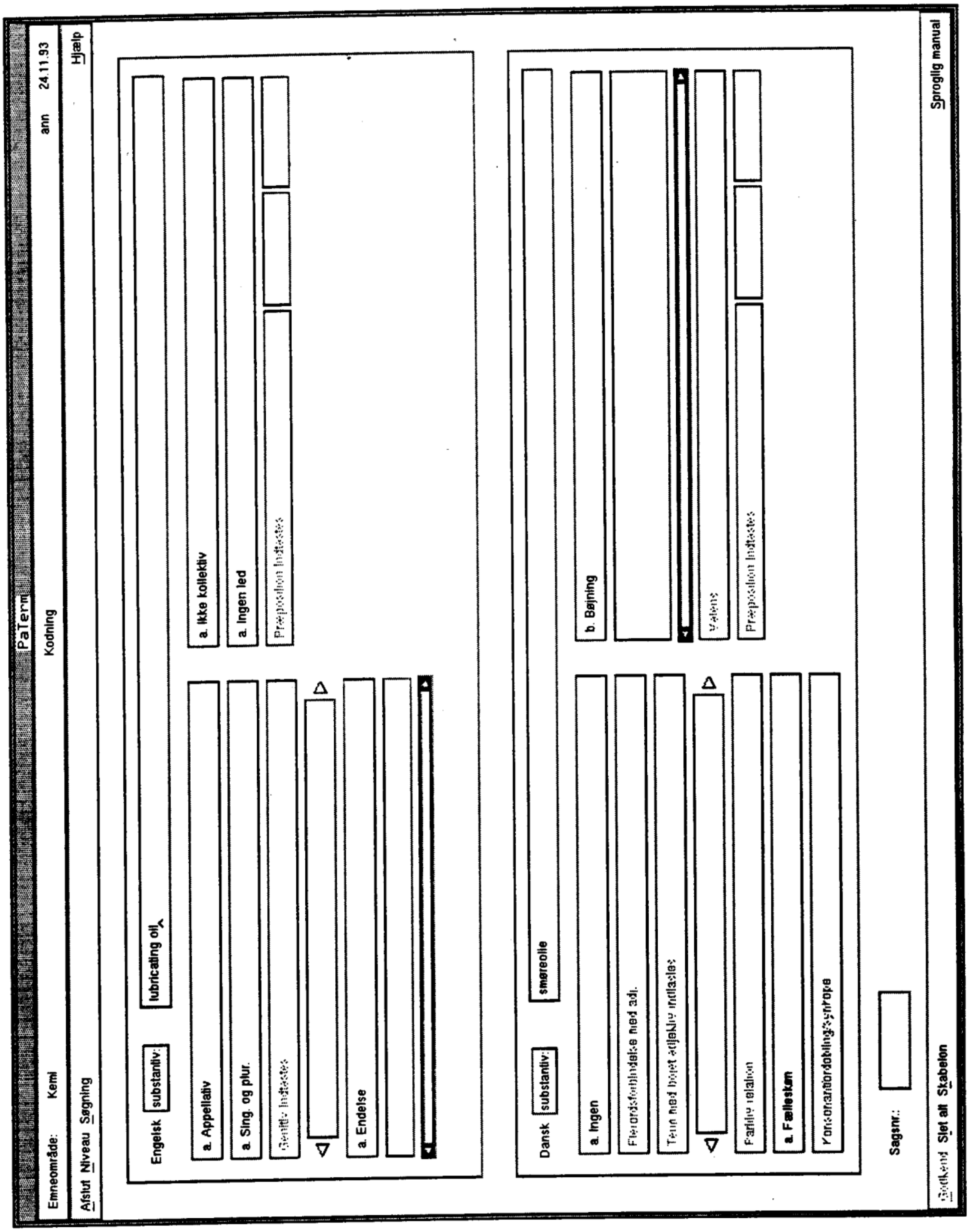


Figure B. Screendump: PaTerm - Term Entry - Expert level.