46

SATISFYING USER NEEDS THROUGH A COMBINATION OF INTERFACE DESIGN TECHNIQUES

*Malin Bladh and Kristina Höök**
SICS, Box 1263, S-164 28 Kista, Sweden
*E-mail: kia@sics.se

KEY WORDS: User needs, user modelling, explanation types, multi-modal interface

ABSTRACT: We show how a combination of careful choice of functionality fitted to users real needs, interface design and stereotypical user modelling can be used to construct a help system for a software development method. The design is such that the *control* of the system remains in the hands of the user, the system is *transparent* to any of its more intelligent parts and it is *learnable*. It constitutes an alternative to user modelling techniques that work in such a way so that their internal workings are not communicated to the user.

1 INTRODUCTION

The goal of the PUSH[1] project is to provide help on a software development method (SDP) for telecommunication applications. Our purpose is to integrate a set of techniques that makes the system adaptive towards the human user, in such a way that the *control* of the system remains in the hands of the user, that the system is *transparent* to any of its more intelligent parts and that it is *learnable*. The advanced techniques that we use are stereotypical user modeling, plan inference and what we call "mumbling". "Mumbling" is part of the scheme of communicating the limits of the system to the user, see [Karlgren 94]. Plan inference is used to increase the understanding of user questions; the questions are interpreted with respect to the task the user is performing [Wærn 94]. Finally, the sterotypical user modelling is realized through the combination of careful choice of functionality fitted to users real needs, interface design and stereotypical user modelling, and that is what we describe in here.

We gathered information on how user's perceive and understand SDP in what we call a *knowledge acquisition* phase. During this phase we found that users of SDP differed in many respects. The user groups were different in terms of level of expertise as well as their roles and general cognitive characteristics.

We could imagine many solutions where we could make choices of explanation patterns and presentation on behalf of the user, but as has been showed before, [Meyer 94, Berry and Broadbent 86], it may well be a bad solution since the user feels out of control. Some of the choices the system could make, as which kind of explanation we should provide, can equally well be made by the user, if the choice is presented in an understandable way. What "understandable" is, we believe to be a combination of domain-dependent solutions, and hiding too complex reasoning in an understandable surface. We use the metaphor "the black box inside the glass box" to illustrate our intention [du Boulay et al. 81]. we hide the low-level system mechanisms from the user, the black-box, and provide some insight into the mechanistics through the glass box.

We also believe that for information seeking domains, it is crucial that the users get the same information back whenever they pose the same question as during previous interactions, what we call learnability. This is especially true in the case of retrieving information on a method which is the backbone of the users daily work. Learnability is also taken to be a communication with the user where the limits and potentials of the system is made clear to the user.

2 PUSH BACKGROUND

The application studied in PUSH is provided by EUA. It is a development process used for aiding the development of new (big) broadband software applications. The development process is available through a huge information space with textual docu-

1 PUSH: Plan- and User Sensitive Help. The project is funded jointly by Ellemtel Utvecklings AB (EUA), SICS and NUTEK. Ellemtel, SICS, Stockholm University and Linköping University participate in the project.

K. Nordby et al. (eds.), *Human-Computer Interaction*

mentation describing it. Since the information space is so large, users are overwhelmed. The purpose of PUSH is to create a help system that will improve this situation.

SDP is an object oriented method. It consists of processes (activities done during the project phases) and objects (specification, code, etc. produced as a result of using the method). Processes and objects are related, objects are related to objects and processes to processes, which is why graphs is a natural presentation of certain aspects of SDP. Other aspects, as definitions and motivations are best presented as text.

Currently, we have a prototype system implemented that exhibits some of the ideas presented below.

3 USER NEEDS ON SDP

3.1 Knowledge Acquisition

The knowledge acquisition in PUSH has been extensive. It has ranged from informal interviews with users, gathering questions from users and constructing answers to these questions, to initial evaluation of our prototype system during a rapid prototype cycle. Our method for gathering specific questions and answers, which is the basis for the work presented here, was divided into:

1 Getting questions on SDP from users - with the special aim of getting the help needs they have in their daily work situation.
2 Finding answers to the questions by the help of SDP developers.
3 Getting feedback on the answers from the users.

During this phase of the knowledge acquisition we have been in contact with more than 20 users, who have provided us with about 60 questions. For about 15 of the questions we have constructed answers which we have taken back to the users in order to get feedback on how usable they are.

From these questions and reactions to the answers, users needs were analyzed into a task structure, and a set of user groups with different characteristics.

3.2 Task Structure

Our task structure describes the users *information seeking needs* which is a small part of the total problem of using SDP, see figure 1. The total problem also consists in interpreting the method so that it can be used for a specific application.

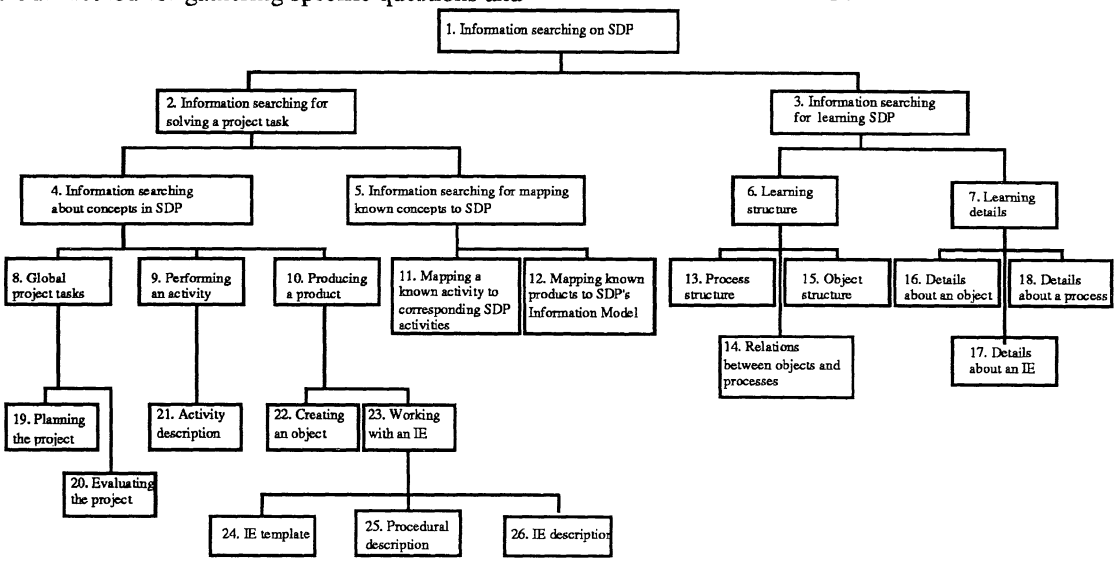

Figure 1. Parts of the task structure of the information needs on SDP.

The task structure reflects both the learning needs (task three, "Information searching for learning SDP"), and information needs that support the work situation more directly (task two, "Information searching for solving a project task"). Among the tasks that support the work, the division between task nine, "Performing an activity", and task ten, "Producing a product", is important. In the first case, the user is inexperienced and therefore feels a need to follow the exact order of activities in SDP. After having done so for a couple of projects, the users often leave the exact ordering of activities and instead become much more focused on what should be produced. Which means that any tool that provides information based on a too strict reading of the order

of activities will not meet the needs of the users. Again, flexibility on the behalf of the help system and allowing users to make their own choice of what they want to see, is important.

3.3 User Groups

Inspired of what other help, advisor or tutorial system base its adaptation on, we chose to investigate the differences between the users in these three main areas:

- The user's experience in, or knowledge of, relevant areas [e.g. Chin 89, Cawsey 93].
- The user's role in the help situation.
- The user's cognitive style [Benyon 93].

The user's experience varies with respects to their knowledge on SDP, telecommunication and software development in general. This points to a need for designing different explanations directed at users with different expertise in these three areas.

Concerning the roles of users when seeking information on SDP, we found: planners of projects, local experts on SDP, project managers and members of a project. These groups need different information, or information presented from a perspective suitable to their goal or task.

Finally, users differed with respect to their cognitive preferences and abilities. One group seemed to like and prefer reading graphs in order to understand how SDP works, while another group shunned the graphs and preferred using search questions in order to find the right information. As pointed out previously, the SDP method contains many relations between processes and objects, and so in order to meet the demands from these two groups, a multi-modal presentation is required, in which navigation is allowed both in the graphical mode as well as through textual questions.

Furthermore, we found that some users know exactly what to ask since they have a good understanding of the problem or of SDP, while others do not know what to ask at all, since they have too little knowledge in order to formulate any specific question. This means that the user must be allowed to pose both explicit questions and *vague questions*.

Coming back to the task structure described above, we can relate the different user groups to different parts of the structure. For example, users with little knowledge on SDP are mostly found performing the tasks related to learning SDP, i.e. "Learning structure" and sometimes "Learning details". Not only the experience level of the user is related to the task structure, but also the roles of the users: for example, the project planners need to perform the tasks related to "Global project tasks".

4 SATISFYING USER NEEDS

In the previous section we outlined some requirements on the design of our help system:

- it needs to allow for free navigation to meet the needs of inexperienced users who are about to learn SDP
- it needs to allow for specific search questions to meet the more experienced users, and users who find it hard to interpret and navigate in graphical structures,
- it needs to allow for vague search questions to cater for users who cannot find the right information through using graphs, but who have too little knowledge to formulate specific well-defined search questions,
- the answers provided to a user must differ in terms of the users expertise but also to meet users acting in different roles,
- finally, the help system must, of course, meet the real needs of users in terms of what users ask about, rather than what developers of SDP think that users may ask about.

It could seem as that the obvious choice of solution would be an adaptive help system which could change its whole interaction with the user based on its inferred knowledge of the user. Instead we emphasize the importance of making the system transparent and leaving the control to the user. Systems that act on their own, without allowing the user control are not accepted by users [Meyer 94, Berry and Broadbent 86]. Below we show how this is realized through which questions we allow, which explanations are possible, and how the system uses multimodality both for input as well as output.

4.1 QTs, EUs and ETs

Most importantly, the help system must contain the information users need, and it must be accessible in ways that correlate with the kind of questions users typically ask. Our approach was to do two kinds of analysis and structuring in parallel, using the data we had collected:

- we identified a set of questions types (QTs) from the real users questions that we had gathered, for example, "Describe X", "Compare X to Y", etc.,
- we structured the answers into explanation units (EUs) that corresponded to the various parts of an answer needed, as a general description of an object, the purpose of having that object, the relation

to other objects, a procedural description of how to generate such an object, etc.

We iterated over this division several times, both with the purpose of relating the QTs to the EUs (not a one-to-one mapping, but rather a one-to-many mapping), and also with the purpose of finding the influence of the users characteristics on the interpretation of the QTs and choice of EUs. This led us to introduce explanation types (ETs), which are used in order to meet the demands from different user groups.

We have divided the ETs into:

- *simple*: high-level explanations, (for novices)
- *detailed*: explanations with details, (for novices trying to understand SDP in more detail)
- *advanced*: in-depth explanation, presupposing an understanding of SDP, these explanations will not avoid the hairy, gory details, (for project members working with SDP producing project results)
- *telecom-related*: in-depth explanations, expressed in terms used by telecommunication experts (directed at meeting the needs of users who are experienced in other methods for producing telecommunications systems).

The about 10 QTs were compiled from the about 60 questions we have gathered from users. The questions are both specific questions, and more vaguely expressed questions[2]:

```
"Does the R object or the O object associated with
the M object appear in the U object?
"Wants to check the instruction of the process X?"
```

4.3 Posing Questions

In the PUSH prototype, the users pose their questions through either choosing among a set of predefined, specific, QTs, or they can pose the questions in free natural language format. The reason for having both, is that the predefined questions help the users to formulate their problems in something that the system can understand. The pre-defined questions communicate the potential of the system, and forms an alternative navigation form to be used instead of clicking in the graphs.

This gives the QTs a natural situatedness, thereby avoiding several otherwise potentially troublesome alternatives for interpretation. This means that not all questions will be accessible from the top level of the system. At the top-level of the system, a set of general QTs are available as pre-defined questions.

[2] Due to secrecy reasons the real process and object names are made anonymous.

The free natural language questions, allow the users to search the database in whatever order they wish to, instead of following the structure imposed by the help system. When a user poses a query in free form rather than using the QTs, it is quite possible that the query will be difficult, or indeed impossible to interpret for the system. However, a user should not have to experience that the system is unable to provide any answer, and so we make sure that the system always answers something to a question. This means that the free questions can express vague information needs, and the user can get help with reformulating those needs into questions which will get at the right explanations (this is what we mean by "mumbling" see [Karlgren 94]).

The predefined questions are divided into high-level general questions, as for example, "Describe process X", and more specific, context-dependent, follow-up questions, as "How is the object X used in this particular process Y?".

The follow-up questions are displayed as pop-up menu's associated with mouse sensitive high-lighted words in the explanations, so-called "hot-words" [Kobsa et al. 94], and they contribute to three aspects: one is to help the users specify their vague information needs, a second is to allow the users to be unhappy with the explanation they are provided with and alter it, and a third effect is that the users are provided with hints as to what they should be asking in this particular context, a form of tutoring or learnability.

An important part of our knowledge representation is to allow for rhetorical links between concepts in SDP that are related only due to explanation reasons. For example, two processes in SDP, X and Y, are quite similar, and so users often ask about the difference between the two. This comparison is not necessary between all processes. Rhetorical links are then displayed as contextual dependent follow-up questions.

4.4 Knowledge Representation

We have chosen to represent SDP with an object-oriented representation - a quite natural choice for an object-oriented method.

It is crucial that a knowledge representation of something as complex as SDP, and where SDP is a method under development with recurring releases, is done in such a way that it is easy to maintain. We therefore employ the principle that any information item describing some SDP concept, should only be described once.

Our representation of SDP is by the object-oriented description, but to fulfill other needs as, for example, on the purpose of a certain process or object, we have extended the representation with canned texts. These texts contains the "hotwords", which in turn are connected to potential follow-up questions and their explanations.

4.5 Interface Design

The interface (under development) is divided into two parts: one allows the user to pose questions in a textual mode, the other part is a graphical view. See figure 2. The graphical part allows the user to navigate and browse around - this fulfills the need in the task "Learning structure" (see above), and it allows for the vague questions, etc.

The dialogue history is a way of keeping focus, so that follow-up questions are associated with the original question and its context. It provides a means for communicating to the user that the system keeps track of previous dialogue and referents.

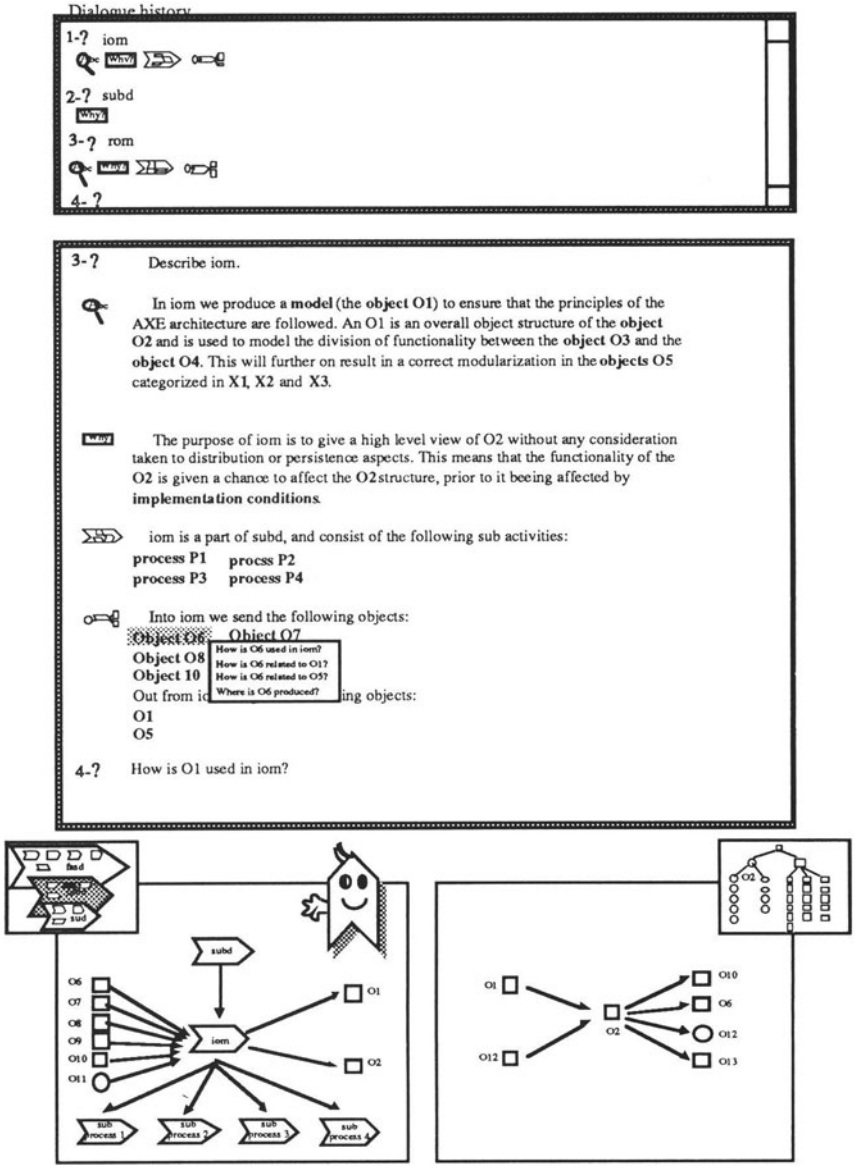

Figure 2. The interface of the prototype help system under development in the PUSH project.

The "fish-figure" pointing out facts in the graphs, is also used to indicate which ET is currently used, see figure 3. This is a means for leaving the user in control - a form of transparency.

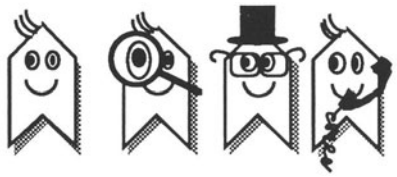

Figure 3. The "fish-figure" illustrating different explanation modes: simple, detailed, advanced and finally telecom-related explanation types.

4.6 Transparency and Control

The questions, follow-up questions, dialogue history and the "little fish-figure" illustrating the explanation mode, are all part in communicating the inner workings of the help system to the user, the glass-box level. They are also aimed at giving the user control of the dialogue. Instead of guessing which explanation mode we should be using for a specific user, we allow the user to make the choice, and the choice is then reinforced by the continuous visual feedback that the fish-figure constitutes. The QTs, and especially the follow-up questions are adapted to the situation in which they occur, and chosen after collecting real questions from users with varying backgrounds and acting in different roles. But through allowing any kind of questions through the natural language input alternative, and also by the sheer fact that the choice of question is all up to the user, the control is still left in the hands of the user.

5 DISCUSSION

Rather than including a user model in the system, we have intertwined the system design with the user modelling done during knowledge acquisition, influencing everything from the choice of functionality, choice of explanations, etc. to interaction modalities. The main objection against our approach could be that it is hard to change the user model if it turns out be wrong, or if it needs to be updated. Through our choice of an object-oriented representation, using the principle that any information item should not be represented more than once (or any links should be made explicit), we try to compensate for the fact that the user model is spread all over the database.

We have shunned any adaptivity made automatic by the system. Still, it remains to be seen whether our solution has solved the whole problem of information overload. In our interviews, we have seen that if the amount of information presented in the explanations exceed one page of text, users will skip it completely. For our test cases, we know that we can stay within the limits of one page of information, but once we extend our database to cover the whole of SDP, this situation may change.

REFERENCES

Benyon, D. & Murray, D. (1993) Developing Adaptive Systems to Fit Individual Aptitudes, *Proceedings of IWIUI*, Orlando, 1993.

Berry, D.C. and Broadbent, D.E. (1986) Expert systems and the man machine interface: part two: The user interface, *Expert systems: The International Journal of Knowledge Engineering*, 4, 1986.

du Boulay, B. O'Shea, T. & Monk, J., The Black Box inside the Glass Box: Presenting Computing Concepts to Novices, *IJMMS*, 14:237-249, 1981.

Cawsey, A. (1993) User Modelling in Interactive Explanations, *User Modeling and User-Adapted Interaction*, 3: 221-247, 1993.

Chin, D. (1989) KNOME: Modeling What the User Knows in UC, In: A. Kobsa and W. Wahlster (eds) *UM in Dialog Systems*, Springer-Verlag, pp. 74-107, 1989.

Karlgren, J. (1994) Mumbling - User-Driven Cooperative Interaction, SICS Technical Report, T94:01, ISSN 1100-3154, 1994.

Karlgren, J. Höök, K. Lantz, A. Palme, J. & Pargman, D. (1994) The Glass Box User Model for Filtering, *Fourth Int. Conference on UM* Hyannis, 1994.

Kobsa, A., Müller, D. & Nill, A. (1994) KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS, *Fourth Int. Conference on UM* Hyannis, MA, 1994.

Meyer, B. (1994) Adaptive Performance Support: User Acceptance of a Self-Adapting System, *Fourth International Conference on UM* Hyannis, MA, 1994.

Wærn, A. (1994) Cooperative Enrichment and Reactive Plan Inference - applying plan inference outside Natural Language Dialog, SIG meeting at *Fourth Int. Conference on UM ,Hyannis*, 1994.