Adaption to the User's Task

by

Kristina Höök
ADAPTATION TO THE USER'S TASK
Kristina Höök
SICS, Box 1263, S-164 28 Kista, Sweden
E-mail: kia@sics.se, http://www.sics.se/~kia
Fax: +46-(0)8-751 72 30, Phone: +46-(0)8-752 15 17

KEY WORDS: User modelling, adaptive explanations, task analysis

ABSTRACT
Adapting explanations to users with varying background knowledge and abilities is a difficult task: the explanation content, style, amount of details, terms used, etc. may be affected in various ways. We have used our analysis of the information seeking tasks of the users in one particular domain as a basis for adaptation. We structured the domain information into a set of information entities where each entity describes one aspect of a node in the information space. Each information entity is fitted to one or several information seeking tasks, and by combining entities we create an explanation adapted to the user's current task.

We do not avoid concepts which are unknown to the user in our information entities. Instead we allow the users to ask follow-up questions on those concepts in order to cater the users' differences in background knowledge. Which follow-up questions are available also depends on the users' current task.

Finally, we emphasise the need to make the difference between the adapted explanations obvious to the user. Only then can the users predict which explanations best fit their need and thereby control the self-adaptive mechanisms of the system.

So, our system is adaptive to the information seeking task of the user, while the user's knowledge, abilities and roles, are catered for by other means.

1 INTRODUCTION
Adapting explanations in an information system can be based on various aspects of the users knowledge or abilities. To this spectrum, we add the users information seeking task as a basis for adaptation. By integrating the adaptivity with a multimodal, interactive, interface, we cater for several aspects of the users knowledge and abilities.
We have studied the needs and problems of a heterogeneous group of users exposed to a large information space as a part of their daily work situation [Höök et al. 1995a, Bladh and Höök 1995, Höök et al. 1995b]. In the domain it is crucial that the users can learn the structure and interrelations of the nodes in the information space, and get some understanding of each node in that space. Since the information space is so large, the users have problems both with navigation and information extraction. Apart from the learning needs, there are also several different situations where users need very detailed and specific information on some particular node. This set of different demands on the navigation and information extraction has caused us to develop an adaptive hypermedia solution which helps the users both to explore the structure of the domain and also to find detailed information.

As we move from written explanations on paper to systems which can systematically aid us in changing the explanations to fit particular users, it is necessary to find which changes will render the users better and substantially different explanations. Adapting explanations to users with varying background knowledge and abilities is a difficult task: the explanation content, style, amount of details, terms used, etc. may be affected in various ways.

Whether one explanation is better than another can only be evaluated against the tasks the user is trying to complete, which is why we have chosen to make our system adaptive to the users' information seeking tasks. Most important is the information content and quality; style and terms used can improve the situation but are, in most cases, only a way of reinforcing the underlying message conveyed by the explanation.

The reason that explanations should be substantially different also on the syntactical level, is that the users will stand a better chance of switching between them and deciding which best fits their current task. If, on the other hand, the adaptations are changed automatically, without the user being in control, or being able to predict and understand why and how the explanations are changed, we may very well increase the information overload problem. Headers, style and terms used may very well be good markers that can help the user to quickly understand which purpose a certain explanation has.

There are two strands in how to approach explanation generation. One is basically centred around the organisation of the knowledge database so that it is possible to generate better explanations [Neches et al. 1985, Swartout and Moore 1993]. The other is to improve the explanation process, i.e. the dialogue with the user or choice of modality for the interaction [Moore 1989]. Our perspective is slightly different from both these directions. We are concerned with marrying a good organisation of the knowledge with the explanation process, since it is the combination of what you say with how you say it that will render good explanations. Our approach is also to divert from the line of research which is directed towards imitating human-human dialogue and explanations. Instead we allow the user, with help from the system, to construct explanations which are fitted to them interactively using metaphors (as direct-manipulation of hypermedia) which are more easily handled by computer systems. This approach we share with other researchers, e.g. [Dahlbäck et al. 1993].
Structure of this presentation

We compare our approach with other adaptive explanation systems, and claim that adapting explanations to the information seeking task may very well subsume adaptations made on the basis of the users knowledge or role. This becomes especially true if the adaptive explanations are paired with allowing the user to manipulate and change the explanation chosen by our system. We also describe an empirical evaluation of whether users are able to connect different explanations with the corresponding task. This is the first step towards making the adaptive mechanisms understandable to the user.

2 OTHER APPROACHES TO ADAPTIVE EXPLANATIONS

Adaptive systems may utilise various kinds of information about the user. An adaptive system may keep information on how much knowledge a particular user or group of users have in the target domain, their current goal with using the system, their cognitive characteristics or other aspects of their personality and style (e.g. [Rich 1984, Benyon and Murray 1993, Kobsa et al. 1994, Chin 1989, Cawsey 1993]).

Based on the information in the user model, these systems can then adapt the content and the amount of information in the explanations, or other aspects of the interaction. Our target system is centered around information retrieval in a large information space, so other approaches to adaptive explanation and navigation are most interesting to study.

2.1 Adapting to the users knowledge

Adaptive systems which keep track of the user's knowledge in a long-term perspective, sometimes over several sessions with the program, can adapt an explanation through:

- avoiding single concepts that are unfamiliar to the user, or by explaining those concepts,
- varying the amount of information, or how detailed an explanation is, [Peter and Rösner 1994],
- providing extra information unknown to the user and not explicitly asked for, but such that the user might benefit from learning it, [Wolz 1993],
- varying the style of the explanation so that users with less knowledge are provided with a more procedural instruction, while experts are provided with a declarative statement, [Paris 1988, Paris 1987, Meyer 1994]

There are also situations in which the explanations consists of graphics or combination of text and graphics, in which case the choice of graphics may be affected by the user’s knowledge [Kobsa et al. 1994].

2.2 Adapting to the users goal or plan

In another line of research, the current goal or plan of the user, rather than their knowledge, is set in focus. If we know which plan a user holds, we can determine whether it is a faulty or sub optimal plan given the goal. So in the system, plans and goals of the whole user population might be stored, while the only information about a particular user is the plan they are
currently executing. Given this information, the system can provide feedback to the user about whether and how the plan is faulty or sub optimal.

Adaptive systems can also hold short-term information about the previous dialogue with a particular user. For example, Giuseppe Carenini and Johanna Moore constructs an explanation through contrasting and/or highlighting similarities with the previous explanation(s) in the dialogue (1993).

Another way of adapting the interface is proposed by Kühme et al. (1992). They propose "adaptive prompting" as a means to provide short-cuts to a set of commands which are most likely to be used given the state the system is in, together with information on what the user usually does in this situation. In the case of adaptive prompting, it is the navigation which is affected by the adaptive mechanisms.

A middle-route between adapting the contents of an explanation and adapting navigation to that information is taken by [Kaplan et al. 1993]. They tie pieces of information in a hyperspace to one another by weighted links. The stronger the weight is, the more relevant are the two information pieces to one another. The weights can also be adjusted to the users behaviour and actions in the system or to group of users preferences.

2.3 Adapting to the users abilities or style

There are also systems which keep information about the users abilities or style. The most notable example is Benyon and Murray (1993) who infer user's spatial ability from their interactions with the system. The system bases a choice of interface style on the users' spatial ability combined with the users experience. So, in this case, the explanations are not adapted, but the interface.

2.4 Adapting to the users task

So, as we can see there are many possibilities both on what to keep in the user model, and how to utilise that information in order to adapt the interaction to better fit a particular user.

What we would like to add to this spectrum is the possibility to keep track of the user's information seeking task, and to adapt both the explanations and also the navigation to that task. We still want to cater for how users background knowledge varies and other aspects of the users' characteristics which are not part of the information seeking task, but we choose to cater for those through other means than being adaptive.

According to Karen Sparck-Jones, (1991), modelling the user can be done in a strong sense where characteristics not necessarily relevant to the functional task for which the system is designed are modelled, and modelling in a more restricted sense limited to those characteristics that are relevant to the system's task. What we propose is even more restricted since we only want to model such characteristics that can help us solve really hard problems where non-adaptive solutions fail. In this case, the problem is information overflow. We try to go as far as possible with the direct-manipulation, hypermedia metaphor, and a good structure of the knowledge database. To that we add adaptivity that helps to provide the user with a subset of the information space.
Our approach here follows what has been put forth by John Self in the area of student modelling in intelligent tutoring systems (1988). Self expresses his critique as "don't diagnose what you cannot treat". We can paraphrase this in the area of user modelling as "don't model user characteristics that do not (profoundly) affect interaction", or perhaps even "only model such characteristics of the user which cannot be catered for by other means". I.e. if we can find ways by which a user can control and alter a provided explanation so that it fits with their knowledge just by making the interface interactive and flexible, that is probably better than making the system guess at the users knowledge or other characteristics. A user model can never be anything but a guess (as pointed out by Judy Kay (1994)).

3 USER NEEDS ON SDP - BACKGROUND

Before we show how our adaptive solution works, we need to provide some background information on the target domain and our studies of it.

Our approach to explanations stems from our experiences in a project, named PUSH\(^1\), devoted to developing and testing different adaptive help mechanisms [Bladh and Höök 1995, Höök et al. 1995b]. The adaptive hypermedia system we developed provides help on SDP, which is an object-oriented software development method\(^2\). The method is documented in a large information space: one variant of the method, SDP-TA, is documented by approximately 500 documents, where each document consists of 5 - 20 pages of text and pictures.

The SDP method consists of processes which are activities done during the project phases, and objects which are specifications, codes, etc. produced as a result of the method. Processes and objects are related, objects are related to objects and processes to processes, and graphs describing these relations form a natural presentation of these aspects of SDP. Other aspects, as definitions and motivations are best presented as text.

The system outlined is implemented using SICStus Prolog Objects, C++ and Netscape with Java. The system architecture is further described in [Espinoza and Höök 1996, Höök et al. 1995b].

Our studies of SDP were guided by ideas from the task analysis area, in particular the Cognitive Task Analysis method [Roth and Woods 1989]. In initial interviews, it became obvious that the users had problems with information overload, they had difficulties in finding information, and once they had found information, they had problems in interpreting and making use of the text and graphs they had found.

---

1 PUSH - Plan- and User Sensitive Help.
2 SDP is developed by Ellemtel Utvecklings AB in Sweden.
In a second round of studies, we collected a corpus of questions, and we went beyond the surface level and extracted a hierarchy of information seeking tasks, schematically depicted in figure 1 (described in [Bladh and Höök 1995]).

The task structure reflects both the learning needs, and information needs that support the project work situation. Among the tasks that support the work, the division between "Performing an activity" and "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. A more experienced user, on the other hand, might take a reverse engineering approach. They know what the end product should be, and so they structure their results after having produced them to fit with the demands of the method.

In our studies, we could see that users did indeed vary in terms of how much background knowledge and knowledge on SDP they possessed, and also in their (cognitive) abilities affecting how they made sense of the texts and graphics provided in the on-line manual [Höök et al. 1995c]. They also varied in terms of which roles they had in their projects, which in turn influenced their information needs. Finally, in a short-term sense, they varied in which information seeking task they had when entering the system.

4 STRUCTURE, INTERFACE AND ADAPTIVITY

Our main problem in this particular domain is information overload and with this problem comes navigational problems and difficulties in interpreting the relevant information in the explanations. We tackle these problems by finding a good basic structure of the domain information, by making a careful choice of how to design the interface to the system, and finally, by adapting the explanations and navigation to the users information seeking task.
4.1 Structuring the domain

Given the heterogeneous user group and their demands on information and the search for it, our first concern must be on how to structure the domain information. The following principles guided our approach to structuring the information:

- it must be possible for users to make sure that they have seen all the information about a particular process or object,
- the risk of getting lost-in-hyperspace must be minimised, so it is better to navigate between a smaller set of well-defined bigger nodes, than a larger set of small nodes,
- instead of making the general concepts into their own nodes to which the users must navigate, explanations of such concepts should be made available in the context in which they are used.

We based our structure of the nodes on the domain structure. Each process and object in SDP is made into a node in the information space\(^3\). Since our interface has been implemented in World Wide Web (www), the user will get all the information about a process or object in one (answer) page in www.

Within each such answer page, we structure the information on processes and objects into a set of information entities. An information entity can be a piece of ready-made text, a static picture, or a representation of the processes or objects relations to other processes or objects. The relational information can be used to generate text or graphs, or sometimes both.

An information entity is a piece of information on a process or object which will provide all information about a certain aspect of that process or object. The entity is stand-alone; it is not necessary to read one entity before another and one information entity have none or few relations to other information entities. It has been shown that this is not an impossible requirement on technical documentation. For example, Svenberg, (1995), studied technical manuals, and found that the information could be divided into pieces of text which had none or few references to other parts of the text.

In order not to repeat information in several of the information entities, it is of crucial importance that the entities are chosen so that the describe different aspects of the processes and objects. In the current version of our system, we have approximately 20 information entities which describe different aspects of processes and about 15 which describe aspects of objects. In figure 2, we see a snap-shot of the interface where a process is being described. In the graph, we can see input and output objects, the process superprocess and its activities - the same information is repeated in the text further down in the page (not visible in the figure). In the text-part of the screen, we see the summary information entity, which provides an introduction to the process description, and the purpose entity, which describes the underlying reasons and motivations behind the process. There is also a basic introduction text which is not opened.

\(^3\) There are approximately 15 processes, each consisting of 7 - 15 activities. There are furthermore 82 objects, each consisting of 6 - 8 so-called information elements.
Figure 2. A screendump of the PUSH system.
Most of the information entities are in the form of ready-made texts in our database. Whether those texts should have been generated from a more basic representation rather than kept as ready-made texts in our database, is an interesting question, but one which is beside the point of this paper. The important issue here is the structuring of the information into information entities, not whether that information has been generated or not.

4.2 Adapting the explanations

Our first attempt in catering for users with different background knowledge, was to write each information entity in several different versions; one for novices, one for experts, one for users who had experience of another method previously used in the company, etc. This approach had several drawbacks:

• it quickly became difficult to keep track of all the information pieces and make sure that nothing was repeated or inconsistent,

• SDP is not static, recurrent releases means that the information entities are not written once and for all, but must be easy to change and maintain,

• the authors of the texts must be able to keep different user models in their heads while writing the different versions of the same texts,

• only for some of the information entities were the differences between the versions of the same entity obvious to the reader.

Instead, we decided to take a closer look at our information entities to see how we could cater for the users varying background knowledge and even more importantly, their reasons for reading the text. We found that by adding some information entities we could construct a reasonable answer consisting of a subset of those information entities, provided that we knew which task the user was currently involved in (exactly how is described in section 4.3 below).

We divided the 'example' -entity into two different information entities, one with a simple example and another with an advanced example. A simple example for project planning purposes may be the same example that the user who is learning about SDP needs. It is also easier for the authors of the example to distinguish between how to find and describe a simple versus an advanced example, while writing the same example in several textual versions to fit different user groups is quite difficult, and not very useful. In general, it was easier to write texts with different contents or purposes than to rewrite the same information in several versions.

So the issue now became whether our information seeking tasks covered all the relevant differences between novice and experts users, and users with different roles in their SDP projects? The task structure partly subsumes some of these differences: a project manager will be seeking information to do with "Global projects tasks", a novice will be trying to learn about SDP which is catered for by the "Learning the method" task.

Still there are some differences between novices and experts which are not subsumed by the information seeking task hierarchy. The first is their understanding of more fundamental concepts in SDP (as 'object-oriented analysis' or 'reuse'), which will be poorer than the experts'
understanding. So, in addition to refining the information entities into a slightly larger set of basic information entities, we also introduced hotwords into the text. Hotwords constitute a means of making certain words clickable and possible to pose follow-up questions on. By allowing the user to pose a follow-up question on what a fundamental concept in the text means, the user can turn an explanation with expressions unfamiliar to him into one where all the difficult concepts have been explained. So, rather than avoiding unknown concepts (as proposed by, among others, [Sarner and Carberry 1992]), we place them in their natural context in the text, and then allow the user to ask follow-up questions on them.

Since the choice of which hotwords should be explained is in the hands of the user, we avoid another problem with the difference between novices and experts. Novices knowledge cannot simply be classified as a strict subset of the concepts, as depicted in figure 2a (taken from [Wolz, 1993]). Instead their knowledge is sometime at an experts level, while in other areas it is at the novice level, as depicted in figure 2b, [Wolz, 1993].

![Figure 2a. Traditional view of expert versus novice knowledge.](image1)

![Figure 2b. Novices knowledge may partly be at the level of the experts.](image2)

Another difference between novices and experts is how they can understand and make use of instructions. This difference has been discussed by Cecile Paris, who analysed naturally occurring texts, encyclopaedias, directed at children and directed at adults (1988). She noted that explanations to novices were process-oriented while experts received part-oriented explanations. The same kind of difference is used by Beth Meyer in their system which distinguished between short, declarative instructions and lengthy procedural instructions of how to manage a cash register (1994).

To handle this difference between novices and experts, we made an exception from our idea not to write several versions of the same text and created two information entities for instructions on how to apply SDP:

---

4 We have borrowed the term hotword from [Kobsa et al. 1994] who uses it to denote the hyperlinked words in the text.
• A lengthy, procedural instruction, with many hints on how to complete the task, how to think when doing it, etc. This text is written in a subject-direct and procedural style: "first you do this, then you go on with that, ... ." The information entity is named: "How to work in this process."

• A declarative description of the state that the project development will be in after the activity has been performed. The information entity is named "What is done in this process."

Examples of the procedural and declarative descriptions are shown in figure 3. The explanation are made anonymous due to secrecy reasons posed by Ellemtel Utvecklings AB. The words marked in bold are the hotwords upon which follow-up questions may be asked.

Activity: Adjust the whole object model
Take a step back and view all the object types together as a whole, and revise the object model.

The object model is the collection of all object types. In this activity the designer steps back and views all object types together and their interaction. The goal is to revise the model and verify it against requirements.

The final object model should contain the object types needed to perform the usage cases. The object types should be appropriately specified in terms of relationships and behaviour.

The model should also be studied for inconsistencies and to check that it is complete and nothing has been forgotten. This is best done by studying the object model together with the requirements described in the functional description of the subsystem.

Activity: Adjust the whole object model
The developed ideal object model is studied as whole and revised if necessary.

Figure 3. One procedural and one declarative description of one activity in a process in SDP.

Let us discuss one final difference between users with different information seeking tasks, background knowledge, ability and role, namely how they navigate in the information space. We found that when the user is learning about the structure of SDP or doing reverse engineering, the relations between different objects and process in SDP is crucial information. If in-
formation irrelevant to these tasks is presented at each node, the user runs the risk of getting lost in hyperspace or being lead astray from their original problem.

Users who are learning details about a certain process or object, or who are working with a particular project task have other navigational needs. They must be able to pose questions that single out pieces of precise information. From those nodes it must be possible to ask follow-up questions which help to clarify all details.

Our approach to satisfy the demands on navigation is through allowing multi-modal navigation, and by allowing the user's task to affect which follow-up links are made available on the hotwords. One navigational mode is through clicking in the graphs, and another is by allowing questions in a restricted natural language format. A third mode is via pre-defined questions associated with the hotwords. By clicking on such a hotword and posing a follow-up question, the user will in fact navigate to that node's answer page.

So, in summary, we have created a set of information entities with associated follow-up questions, so that when we know that the user has a certain information seeking task, we are able to compose an explanation of these information entities that is fitted to that task. The information entities differ both in terms of content and also in linguistic style. The navigation between the nodes in the information space is also affected by the user's current task.

In addition to choosing which information entities the user should be presented with, we allow the user to manipulate the answer. The follow-up questions on hotwords will help novices clarify concepts not well understood. The user is also allowed to close whole information entities, or open such entities which the system originally decided should be closed.

4.3 Making explanations fitted to the task

Let us assume that we are able to know which information task a particular user has. Our system can then create an explanation fitted to the task through choosing among the set of information entities and the navigation through the information space may also be affected.

The choice of task together with a question that the user poses will affect which of these information entities that will be shown from the start. Other information entities will be available to the user so they can choose to open parts that are initially closed. The first choice of what is open will give a reasonable amount of information given the task and a particular question.

In figure 4 we give examples of rules, or explanation operators, that control how the question "describe process X" should be answered given a particular task. The information entities in the right-hand side of the rule, are those which will be 'open' initially.

The learning structure task is where we allow the user to "surf around" the information space in order to get a feeling for what different SDP concepts stand for, what is important, how different items are connected, etc. Only very basic information is shown at each node, mostly in graphics. For a process, we shall see a textual basic introduction, a textual description of the underlying purpose behind the purpose, and a simple example. In the graph we shall see which activities the process consists of, which input and output objects the process has, and its relations to sibling-processes.
The *project planning task* provides the project planner with the kind of information needed in order to make decisions about how to work with SDP-TA. Most important is to provide information as to why and when this process should be applied. A summary of what is supposed to happen in each activity in the process provides material for the project planners task. The information model shows the relations between activities and objects. Finally, a simple example helps the project planner to grasp how much work will be needed in order to complete the process.

The *following a process task* is the most elaborate - it helps the project members with detailed information on how to work in a project. They are provided with a textual introduction, followed by a detailed procedural description the process and of each activity. Information on input and output objects is provided both in graphs and in text.

The *producing a product task* stems from our studies which showed that a lot of the time the SDP-TA users will not follow the method as it should be followed. Instead the just go ahead and produce code and documentation which they know will be needed. In the end, they have to turn their results into the correct SDP-TA object structure. They need the information model, which displays the relations between activities and objects. That way they can trace backwards and find the places where the results they have already produced should be placed.

Learning structure →

  Basic introduction, Purpose, List of activities, Input objects, Output objects, Relations to other processes, Simple example

Project planning →

  Project planning information, What is done in this process, Information model, Simple example

Performing an activity →

  Summary, How to work in this process, Release information, Input objects, Output objects, Relations to other processes, Entry criteria, Exit criteria, Information model, Advanced example, Frequently asked questions

Producing a product →

  Information model, What is done in this process, Release

**Figure 4.** Rules for describing the relation between some tasks and information entities for the question "describe process". The information entities are only described by their name in this figure - no full example can be included due to the secrecy agreement with Ellemtel.
How to know the users task

Our approach to adaptivity has been to find a balance between a user-controlled and a self-adaptive system [Höök et al. 1995a, Höök et al 1995b]. According to Oppermann (1994) this middle route is to be preferred since the users must have control over the adaptivity, but they will not spend much time adapting the adaptivity when their main task is really something else. We allow the users to set which task they are working with initially, and then we use plan inference (i.e. inferring the users’ underlying goal from their actions at the system) to update their assumed current task continuously [Wäern 1994]. The user can at any time change the inferred task to some other task, although we limit the set of potential tasks to those which will actually change the explanation in the current situation.

4.4 Being dissatisfied with the explanation

As pointed out by Johanna Moore and Cécile Paris, it is of crucial importance that the users are allowed to be dissatisfied with an explanation, [Moore 1989, Moore and Paris 1992]. Their conclusion is that the system must allow for a dialogue between user and system, where in the end the total dialogue will provide users with the explanations that fulfils their goals.

Our system also allows for a kind of dialogue, but it is not imitating the kind of dialogue that happens between experts and novices. Instead, the users can pose follow-up questions that are associated with the hotwords in the texts or in the graphics, and open or close the information entities that are available as an answer to a question, etc. This way, the user can in fact turn an answer originally directed at somebody learning about SDP into a description fitted for a user trying produce an object or follow a process, or they can turn an explanation mainly fitted for an expert in the domain, into an explanation more appropriate for a novice.

We view this 'dialogue' as an example of how a computer can offer other and different ways to communication, and still reach the same goal as if we had chosen to imitate a more human-human kind of dialogue. The great advantage is of course, that it is not as difficult to implement.

5 WHERE THE EXPLANATIONS DIFFERENT ENOUGH?

We have argued that is important that the users can distinguish between the different explanations. Only then can the user make the coupling between task and explanation and learn how to best utilise the system. In order to find whether users could distinguish between the different explanations and tasks, we tested them on seven users. We provided them with four different explanations and brief descriptions of the four information seeking tasks (based on the rules in figure 4 above). We then asked them to pair the tasks and explanations and motivate

5 Self-adaptive systems are such that the whole adaptive process is done by the system alone: the system initiates, proposes, decides, and executes the adaptive behaviour [Kühme et al. 1992].
why they could/could not be paired. The explanations were provided as an answer to the fairly
general question "Describe the process iom" (iom is a process in SDP).
Out of the seven subjects, five were experts on SDP and two were novices. The experts could
be divided into two groups: those who had taken part in developing SDP (three subjects) and
those who had gained their knowledge from applying SDP in projects (two subjects).
Five subjects did a "correct" pairing of task and explanation, while two subjects mixed the
'planning project' and the 'learning structure' explanation. After adding an information entity
with the goal of telling the project planner why a certain process should be employed in a
project ('Project planning information'), these two subjects found the explanations helpful and
distinguishable from one another.
The subjects all reported that having different explanations on SDP like the ones we pre-
presented, would be very helpful to many groups of users.
In order to really study how well our explanations meet users needs and not only how differ-
ent the explanations are, we need to make a longitudinal study where we can see how users in-
formation seeking needs naturally arise, and whether those are met by our model. An interest-
ing extension is to utilise some form of machine learning to adjust the coupling between task
and information entity over time, an issue that will be explored further in our research.

6 SUMMARY AND DISCUSSION

We have approached the information overflow problem in our domain from a usability per-
spective: how can we reduce the amount of information to a reasonable level and still give the
user control of the interactions with the system? The solution is a mixture of making the ex-
planation process usable and finding a good structure of the information database.
Making the explanation process usable is done through giving the user control of the adaptivi-
ity and also control over the explanations. The user can manipulate explanations by opening
and closing parts of an explanation (the information entities) and also by asking follow-up
questions on concepts.
The structure of the information database is based on information entities. A lot of effort has
gone into finding appropriate information entities; they should be different enough for the user
to recognise their underlying purpose and make sense of the information.
We would also like to point at our approach to combine a multimodal interface in www with
adaptively generated explanations. Through catering for novices versus expert needs through
allowing the user to manipulate the answer, we avoid the problem of keeping a user model of
the users knowledge. In general, we give the adaptivity a natural situatedness in the interface
as an integral part of it. The adaptivity will never remove any information - only hide it from
the users immediate view.
Basing the adaptations of explanations on information seeking tasks, appears to be a reason-
able approach to adapting explanations. Especially if the target domain is as large as the one
presented in here.
Acknowledgement

The author wishes to thank all the anonymous subjects who took part in our studies. The work has been performed in cooperation with SICS, Ellemel Utvecklings AB, Stockholm University and Linköping University in the PUSH (Plan- and User Sensitive Help) project, and has been funded jointly by Ellemel, SICS and NUTEK.

Thanks to Benoit Lemaire, Nils Dahlbäck and Klas Karlgren for valuable input and discussions. Thanks also to all the members of the PUSH project.

REFERENCES


