II

User Interface Modelling - adding usability to use cases

MAGNUS LIF

Center for Human-Computer Studies (CMD), Uppsala University, Sweden.

Abstract

User Interface Modelling (UIM) is basically a method for gathering user requirements that are applicable when designing the user interface of an information system. UIM is to be used as a complement to use case modelling (Jacobson, Christerson, Jonsson, & Övergaard, 1992) in the system development process. A goal model, an actor's model and a work situation model are specified during sessions where the end-users co-operate with software engineers and user interface designers. The goal model is a list of high level goals the users want to achieve. The actor's model is a description of characteristics for each category of users. The work situation model, is a specification of work situations, information objects and actions, and properties of attributes and operations, suitable for the design. UIM does not describe a step-by-step procedure on how to create usable interfaces. Interface design is partially a creative process that can not be completely described with a method. Instead the designer is supported with a substantial model containing the requirements on the interface. This model is created during UIM sessions.

The method has been tested in different development projects at the Swedish National Tax Board. It has shown to provide useful input to the user interface design process.

1. Introduction

When developing a computerised information system it is necessary to understand the users' work and needs. In most systems development projects some kind of structured analysis and design is performed where the users work is described with data flow diagrams, data models, etc. (e.g., DeMarco, 1978). The data flow diagram describes how data should be processed within an organisation. These methods do not give suitable aid for developing the user interface (Floyd, 1986). Today it is common to use object-oriented modelling techniques, e.g. Unified Modelling Language (Booch, Jacobson, Rumbaugh, 1997) where the functionality is described with use cases. "A use case is a complete course of events in the system, seen from the user's perspective." (Jacobson et al., 1992, p. 157). However, an object-oriented methodology does not guarantee a usable interface. The methods are suitable for developing several parts of the information system, but they do not provide sufficient support for the design of the user interface. Instead, these methods invite the designer to create an interface where each function or use case is represented with one window on the screen. Usually, the user has to interact with several such windows in order to complete a task, resulting in a fragmentary interface, with a large amount of windows.

Several methods for task analysis (TA) have been developed to assure that human-computer aspects are introduced in the design process. TA is generally concerned with what people do to get things done (Preece et al., 1994). Most methods for TA include decomposition of task into sub tasks and a sequential description of how they are performed. One problem with methods for TA is that the descriptions of the tasks are too fine-grained (Gulliksen, Lif, Lind, Nygren, & Sandblad, 1997). When designing the interface a description of bigger concatenated tasks are more useful. Benyon (1992) claims that TA is not capable of being independent of the device that is used when carrying out the tasks. It is therefore a great risk that current practices will be embodied in the new system. He means that Human Computer Interaction (HCI) should contribute to systems design with modelling tools to complement existing methods and tools of systems analysis, instead of "inventing wheels which have already been painfully discovered by others". Sutcliff and Wang (1991) mean that HCI principals are poorly spread partly because of lack of integration between HCI research and practice and methods in software engineering.

In this paper, a method for gathering requirements on the interface is presented. Our intention is not to present a whole new method for systems development. Instead we have focused on a small but important part, that is how to gather user requirements applicable when designing the user interface. User interface modelling (UIM) is intended to be used in conjunction with use cases in order to support the designer of the user interface with relevant information.

1.1. The use of use cases

Use case modelling differs from traditional methods for systems analysis in the sense that the functionality of the system is modelled with an object-oriented approach. Object-oriented modelling has become rather common lately mainly because it is advantageous in terms of reusing objects and mapping to the real world. There are several techniques for modelling available. Use case modelling is now a part of the Unified Modelling Language, UML (Booch et al., 1997). UML is a unified version of three different object-oriented modelling techniques. The model is documented in terms of use case diagrams, class diagrams, behaviour diagrams, and implementation diagrams.

In UML the requirements on the functionality of the system are described in terms of *use cases* and *actors*. An actor represents what interacts with the system. There are several actors, both human and non-human, exchanging information with the system. A non-human actor is, for example, another system. In UML an actor is regarded as a kind of class where each instance of such a class is a user. A user can play the role of several actors.

According to the use case approach, users perform work by carrying out sequentially related operations on the system. A use case is a specific way of interacting with the system by performing some part of its functionality. It is a special sequence of related transactions performed by an actor in dialogue with the system, e.g. withdraw cash from an automatic teller machine. A use case is also regarded as a class. Each instance of such a class is in UML defined as a scenario performed by the user. The instance exists as long as the use case is operating.

Muller, Haslwanter, & Dayton (1997) claim that there are some problems with the use case driven approach. One problem is that the use case model usually is written with the software system as the focus of attention. They mean that the use cases give too little priority to the end users and that each use case is a definition of user actions by software engineers. To overcome these problems *it is necessary to model the use cases in participation with the end-users.* Otherwise there is a great risk that the application will not support the users efficiently in their work.

Describing the requirements in terms of actors and use cases gives good support when defining how the involved objects communicate in the system. The users can be involved at an early stage and they are able to describe their work in a terminology that easily can be adopted by both the users and the developers of the system. However, the results from these analyses do not support enough information to create a usable interface why some additional modelling is needed.

1.2. The need to model the user interface

UIM has been developed in co-operation with the Swedish National Tax Board. Initially, we studied how analysis and design were performed traditionally within the organisation. This was performed during system development projects by observing, taking notes and studying modelling results.

Our observations and earlier research experiences motivate why a new method is needed:

- User interface design is usually performed by software engineers with limited HCI knowledge. They seldom have enough time, interest, and experience to be able to create usable interfaces.

- The result of the modelling is not always used by the software engineers. Observations where interfaces are created without studying the modelling results are not uncommon.

- Several design decisions were made during the analysis phase. In several projects the users' work was described in terms of how the user interface should look and behave, e.g. which windows, scroll bars and buttons to include.

- In some projects the users made all the design decisions, even those that were not related to their work. Sometimes this resulted in bad design solutions. The users are experts on their work not in user interface design.

- It is difficult for the users to describe a future work situation when this means significant changes of the work practices.

We have previously stressed the need to bridge the gap between analysis and design (Gulliksen, Lind, Lif, & Sandblad, 1995). Bridging this gap completely can be difficult. However, it can at least be narrowed by:

- involving the users in the development process (e.g. Greenbaum & Kyng, 1991),

- including a user interface designer in the process

- supplying the designer with a substantial model describing characteristics of the users work that are relevant when making design decisions.

An attempt to model the requirements on the user interface based on a use case driven approach has been described by Constantine (1995), through Essential modelling. Three separate models are created: the user role model, the essential use-case model and the context model. The user role model describes characteristics of the users that are relevant for the design. The essential use-case model describes the interaction between the user and the system as generalised scenarios. Finally, the context model, is a collection of abstract user interface elements describing the application. Essential modelling describes the users work in terms of interaction with the system, giving basic guidance on how the user interface should look and behave. Essential modelling is probably a good support when developing smaller systems where the design solutions are rather obvious. However, there are some limitations with this approach. Essential modelling is too system oriented, describing how the system should react on the users' actions, which in turn will limit the design space. Also, it does not give enough guidance on what information the user needs or how the information should be used. For complex development projects we mean that more detailed descriptions of the content of the user's work are needed. Gould, Boies & Lewis (1991) discuss the importance of separating the content of the interface (e.g. the substance) from the style (e.g. the look and feel). The style of the interface should be specified by a designer, supplied by a substantial model describing the content of the user's work.

1.3. The designer in focus

User interface modelling has been developed with the designer in focus. Understanding his or her needs is necessary in order to develop a method that captures the relevant aspects of the users work. So, what is the essence of the designer's work? The main task of the designer is to optimise the user interface based on the different requirements (Figure 1). During this process it is necessary to understand who the end-users are in terms of knowledge, experience, etc. It is also important to understand the domain, i.e. the users' work and needs. Not only which information that is needed but also how the information is used (Gulliksen et al., 1997). The size of the screen, the development tool to be used and other technical aspects are also important constraints. Finally there are rules and recommendations, style guides and guidelines that has to be considered. To be able to make the right design decisions it is necessary for the designer to have all of these requirements at hand. With too little information it is most likely that bad design solutions will occur.

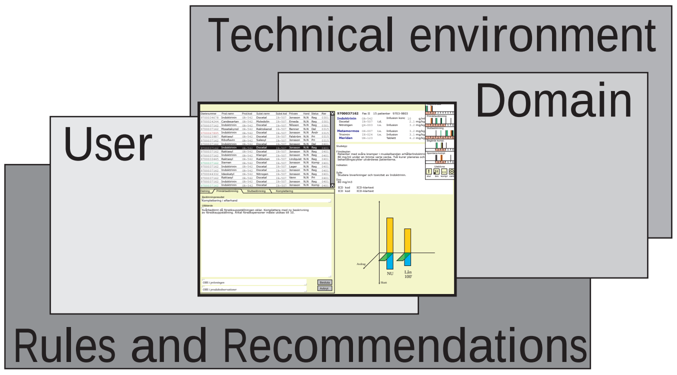

Figure 1. Design is mainly concerned with optimising the user interface based on different requirements.

This paper gives a brief description of use case modelling followed by a detailed description of UIM. The use of the method is illustrated with a system for reporting working hours.

2. Use Case Modelling

In use case modelling, the user requirements are described in terms of actors and use cases. First, the different actors are identified and then the use cases with which each actor communicates. In the use case diagram, the relation between an actor and a use case is shown by connecting the actor with the use case by a solid path. For further reading (c.f. Booch et al., 1997).

In this paper a smaller information system for time reporting is used to exemplify how UIM is to be used. With this system all employees at a company should be able to report how many hours they have spent within different projects. The reports are checked and administrated by a manager.

Example:

In the time reporting system four different actors were identified: *Employee*, *Manager*, *Administrator* and *Agresso*. The first three actors are human and the last actor, *Agresso*, is another system. Each actor communicates with different use cases (Figure 2).

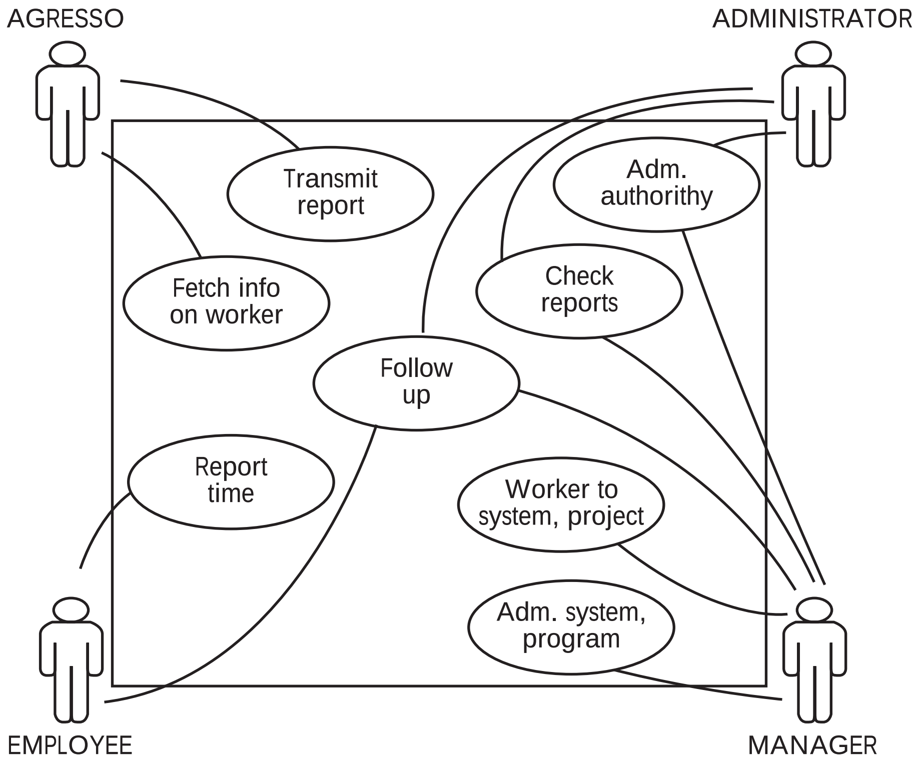

Figure 2. Each actor communicates with a number of use cases.

3. User Interface Modelling

UIM is performed in sessions with users, software engineers and user interface designers. The sessions are led by a modelling leader who is responsible for guiding the discussions. In this process it is necessary to cover all parts of the user's work, using a top-down approach. All participants should be able to contribute with their knowledge. The models are outlined on a white-board, on slides or in a similar way. After each session, the models are documented on paper and distributed to all participants. The models are developed in an iterative process, because the models have to be updated and complemented as the development of the system continues.

Three different models are created, suitable for the design of the interface.

- Goal model
- Actor's model
- Work situation model

3.1. Goal model

When users perform work they have certain goals they want to achieve. Goals can be specified on different levels. In cognitive modelling such as GOMS (Card, Moran, & Newell, 1983) goals are usually specified on a low level, e.g. enter a character. In UIM the goals are specified on a much higher level, e.g. products must be delivered within 24 hours. For the designer it is essential to know which these goals are. They are valuable when judging different design options. Goals, as defined here, are similar to *criteria* as defined in Design Rationale (MacLean, Young, Bellotti, & Moran, 1991). However, they differ in the sense that in UIM each goal is given a priority. The designer has to know which goal that is most important since it usually is difficult to fulfil all goals.

Each goal is described from the users point of view. A good practice when defining the goals is to first make a long list and then keep maybe the two or three most urgent ones. Each goal is then given a priority.

The goals are documented as a list in free text, with the most urgent goal first. The most important goal has priority 1, the second most important has priority 2, etc.

Example:

In our example a time reporting system is developed. The goals the users wanted to achieve where the following:

1. Ease of use. All personal, including consultants, must be able to quickly report their working hours every week.
2. It should be possible to report time for a whole week simultaneously.

3.2. Actor s model

According to the use case approach an actor can be both human, and non-human, e.g. another computer system. In use case modelling the actors are not treated as parts of the system and are therefore not described in detail. When designing the user interface the human actors are of utmost importance. It is essential to know who the users of the system are (ISO, 1995). Therefore each human actor is described in detail. The user interface designer is only concerned with the interface between the human actors and the system. The non-human actors are therefore not further described in UIM.

A detailed description is made for each human actor that has been identified. Each actor is a category of users. Each user in such a category is of course different from the others. It is therefore not always obvious how to characterise an actor, representing a whole group. To solve this problem it is sometimes better to describe more than one category of users for a particular actor, e.g. skilled and novice.

Each actor is documented with a name followed by a description in free text. Important aspects that should be described are:

- Position at the office
- Product experience
- Task experience
- Frequency of use

Example:

EMPLOYEE

- All employed personal report their working hours via the system
- They have no product experience
- All users have previously reported their working hours on paper
- Most users will report their working hours one or two times a week

3.3. Work situation model

It is important that the user always has all needed information and tools simultaneously present on the screen while performing a work task. According to the use case driven approach each actor communicates with a use case through a unique interface. Since a user often has to communicate with several use cases while performing a task it would not be suitable to represent every such interface with a window on the screen. Rearranging windows is time consuming and hinders the user from performing his or her actual work. To help preventing this problem the concept of work situation is introduced.

A work situation is defined as *"a set of related work tasks without sequential restrictions, but with a natural belonging performed in total by one person... One work situation may include one or several work tasks. One actor can handle one or several work situations"* (Gulliksen et al., 1997, p. 282). A user is usually responsible for certain parts of the work performed within an organisation. A work situation can be described as a core work task or a major responsibility. Typical work situations for a sales man can be "Selling products to customers" and "Creating budgets". When the user performs work in such a work situation he communicates with different use cases. Each work situation may include one or several use cases. Sometimes a use case can be represented in more than one work situation. It is important that the user can complete each instance of a use case, in one work situation. Such an instance may include the following actions: search for a certain product; read the proper information; answer the customer's question.

The different work situations are usually identified through the actors. The following questions are relevant when identifying the work situations.

- Which are the major, high level, tasks for each actor?
- Which use cases are logically connected?
- Which use cases are connected in time?
- Which use cases requires access to the same information?

Usually an actor works with few work situations.

Each work situation is represented with a rectangle with rounded corners in the documentation (Figure 3). Arrows show which work situations each actor may handle, and which use cases that can be reached in each work situation. If two actors are admitted to act in the same work situation, it is possible to tell their different authorities from the use case diagram.

Example:

In the Time reporting system three work situations have been identified: *Reporting time*, *Checking reports* and *Administration* (Figure 3). Only human actors are included in this model.

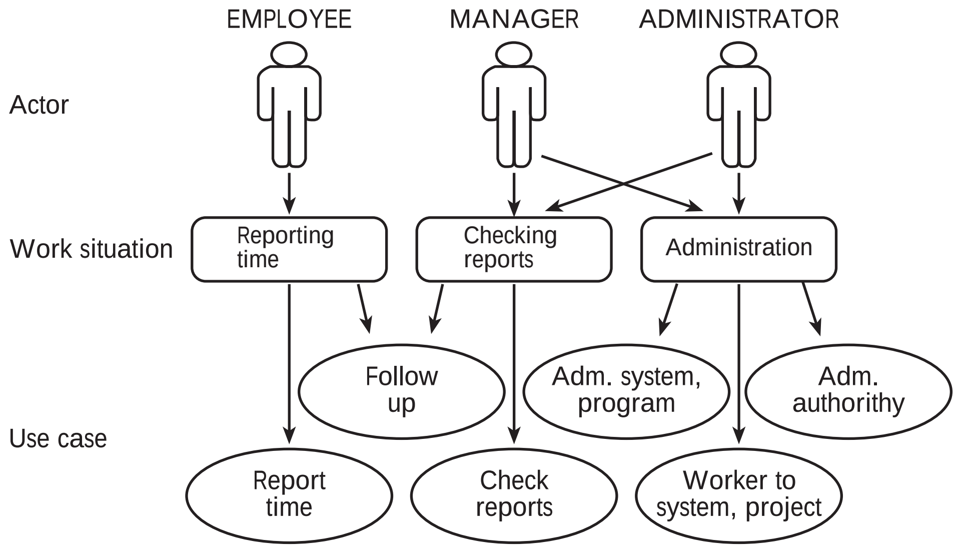

Figure 3. An actor performs work in different work situations. In each work situation the actor can communicate with one or several use cases.

The model shows which work situations each actor may handle and which use cases each actor may communicate with in a work situation. It provides an overview of the users work. It is important to note that no design decisions have been made yet. The model describes which use cases that have a natural belonging; not the layout or the behaviour of the interface.

Information objects and actions

A user performing work uses sets of information. In the physical world this information may be found in books or forms. In computer supported work a lot of the information is stored in data bases and is made available for the user on the screen. The user uses this information for reading, writing, searching, etc. When designing the interface it is important to know which sets of information the user needs and how the information is used. In UIM such a set of information is called an information object.

An actor communicates with one or several use cases in each work situation. When the actor communicates with a use case he requires different information objects, such as a form. It is important that all information objects are available for the user in a work situation. The user needs different tools and information depending on the work to be performed. The user's requirements on information and functionality are here described in terms of information objects and actions.

When identifying the required information objects a relatively complete data model is needed. In UML this model is called a class model. In the class model, the different classes and their relations are specified. The different information objects needed in each work situation can usually be found in the class model. During this process each use case in a work situation is studied, and the information objects required by the users are specified. Sometimes it is necessary to have more than one information object of the same kind available, e.g. when comparing current with historic information.

The functionality is here briefly described as actions performed by the user when communicating with a use case. The response from the system is not modelled since that could define how the interaction should be performed in the user interface and thereby limit the design space. The actions are identified on a high level, e.g. not simple key pressings.

Each information object is represented by a rectangle in the documentation. If more information objects of the same kind are required, two rectangles are drawn on top of each other. Actions are documented for each use case in free text.

Example:

A model is made for the use cases *Report time* and *Follow up* within the work situation *REPORTING TIME* (Figure 4). The information objects needed when communicating with the use case *Report time* were *Person*, *Report day*, *Report week* and *Project*. When performing this use case the users needed one information object Person and several of the other information objects. They wanted to register working hours for at least one week at the time, so several *Report day* objects were required. They also wanted to see the status for older reports.

Actions for the two use cases are also documented.

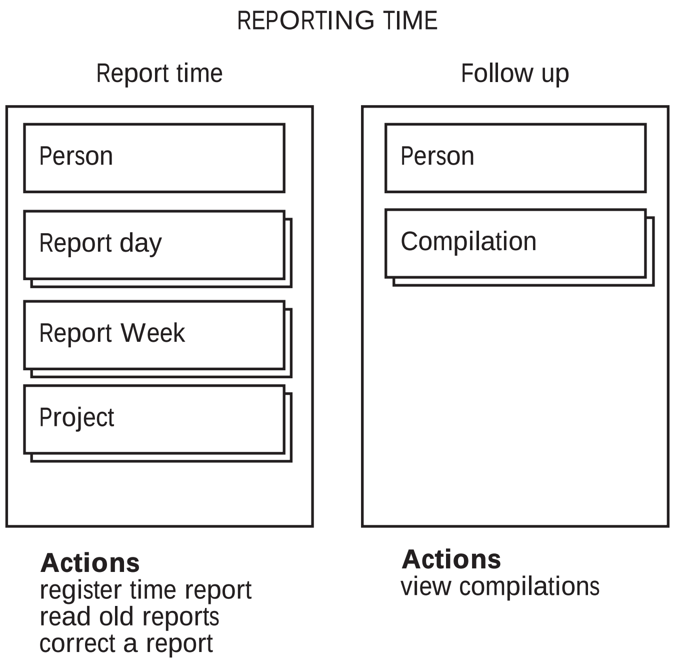

Figure 4. Information objects and actions are specified for each use case in the work situation Reporting time.

This model provides an overview of the information needed in each work situation. By studying this model, it is possible to identify which information objects that are needed in several use cases.

Number of objects, attributes and operations

So far an overview of the users' requirements on the system has been outlined. In this phase a deeper and more detailed description is made. The documentation is more closely related to the class model. The class model defines which attributes and operations each class contains. However, several properties, usually not defined in the data model, are essential when making design decisions concerning the user interface. If such properties of attributes and operations are carefully considered while designing the interface the system can become a more effective support for the user.

Number of objects
Sometimes the user needs to view information from several information objects of the same kind simultaneously. For the designer, it is valuable to know how many such objects that a user needs when performing a task. This may for instance help the designer to decide how many rows in a list to present on the screen.

The number of objects is documented in a diagram (See Table 1).

Attributes
The screen space is usually too limited to hold all needed information at the same time. If possible, all information needed for a decision should be visible simultaneously. If that is not possible some information has to be hidden. To be able to judge which information that is important for the user and which information that is not, each attribute is given a certain *priority*. Information with a high priority should be visible at all times. Information with a low priority may be hidden if there is not enough space on the screen. By giving each attribute a priority it is easier for the designer to decide which information to put in the foreground and which to put in the background. Some information is usually regarded as extra important by the user. Such information can be highlighted in the user interface by using a visual cue, e.g. font, shape, colour.

Other things that are important are *default values* and *task related status*. One example of status can be illustrated with a clinical thermometer. If the temperature is higher than a certain value it indicates that the patient has fever. The status can be either fever or *not* fever. This kind of information can be very important for a user performing a task and is therefore necessary to show on the screen. Status information can also be highlighted by using a visual cue.

Priority is documented in the diagram for each attribute. Important information should be given a priority 1 and less important information a priority 2. If a certain attribute is extra important it should be given a priority X.

Default values, task related status and other things of interest are documented in the column labelled “Other.”

Operations
Some characteristics for the operations are important as well. It is valuable to know which operations that are utilised by the user. Some operations are used more often than others. Frequently used operations should be possible to access by using a short-cut. To help the designer to decide when to use short-cuts, the operations are given different priorities.

Priorities for the operations are also documented in the diagram. Frequently needed operations should be given priority 1. Otherwise they should be given a priority 2. Other things of interest are documented in the column labelled "Other."

Example:

Work situation: *Reporting time*
Use case: *Report time*

Information objects	**Number of inform. objects**	**Attribute**	**Priority**	**Other**	**Operation**	**Priority**	**Other**
Person							
		pers. ID	2		Change address	2	
		Name	2				
		Tfn.no	2				
Project	All types						
		Type	1				
Report Week	52 (1 year)						
		Week no	1	Default: current	Copy old report	1	Default: last report
		Status	X				
Report Day	5 (one week)						
		Day	1				
		Hours	1				
		Note	2				

Table 1: Characteristics of attributes and operations in a work situation are documented in the diagram.

3.4. Example: The design of the user interface

Three models describing the users' requirements on the interface have been created. The example illustrates how these models can be used in the design process.

Figure 5 shows a view from the time reporting system. The primary goal was that the system should be easy to use; everyone should be able to report their working hours quickly. The second goal was that it should be possible to report time for a whole week simultaneously. According to the actor's model, the employees will report their working hours one or two times a week. In the prototype a calendar is used as a metaphor for reporting time.

In the work situation model, three work situations have been specified. In the prototype each work situation corresponds to one screen sized workspace. The figure shows the workspace corresponding to the work situation *Reporting time*. The user may select workspace via a panel that can be visualised upon demand.

For this work situation some different information objects and actions were identified. This model is especially useful in the beginning of the design process since it enables the designer to get a "bird's-eye view" of each workspace, making it easier to get a grip of the entire application and also to identify information objects and actions that are common for several use cases.

More detailed information on the different attributes and operations were also identified. On the left side of the screen all types of projects are shown. The calendar shows 5 days at the time, as defined in the diagram. In the lower area of the screen, week number and status are shown for each of the 52 weeks. Three different colours are used for showing the status of each week. The number of hours that a user reports is drawn with a pencil. The colour of the pencil indicates which type of activity the user has performed within a project. In the prototype it is also possible to see how many hours that have been reported for each day, for each project and for the whole week. This corresponds to the object *Compilation* in the use case *Follow up*.

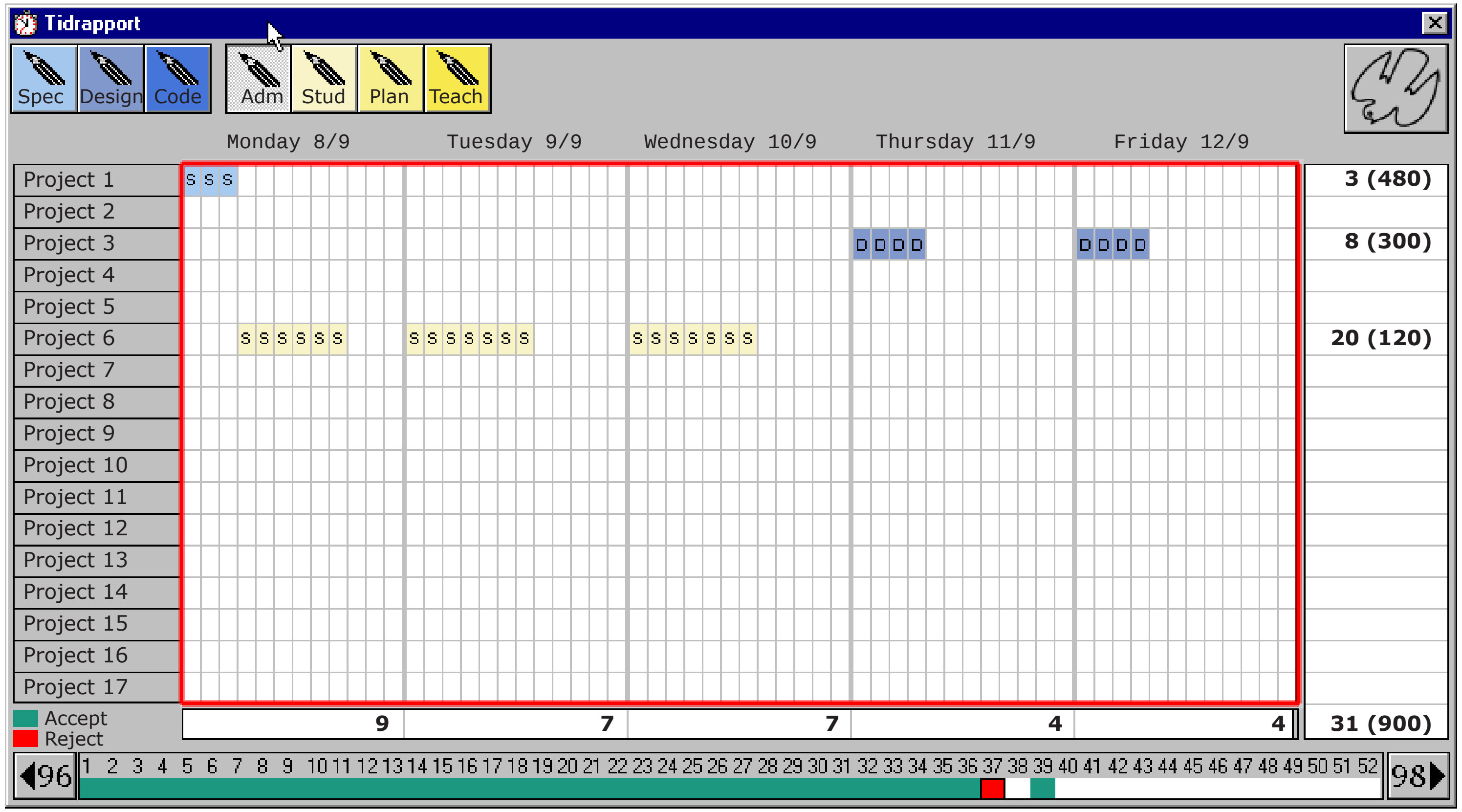

Figure 5. A design example. The time reporting system. (Colour plate 2).

4. Evaluation of the method

User Interface Modelling has been developed in co-operation with the Swedish National Tax Board (RSV). During the development process we have adopted several methods in order to receive necessary background knowledge and to develop and evaluate the method.

4.1. Action research

In action research the researcher is not an independent observer, instead he takes an active part in the process in order to identify new solutions to a defined problem (Rapoport, 1970). We have been involved in several systems development projects at RSV where we have functioned as HCI experts and researchers. In the role of HCI experts we have been giving advise concerning the design of the interfaces and identified shortcomings in the analysis methods traditionally used within the company. We have also been able to present and try out new ideas on how to gather user requirements on the interface. UIM was developed in an iterative process during approximately 2 years and it has been used in different projects at RSV. In two of these projects we were more deeply involved which enabled us to get feedback on the utility of the method by participating in meetings concerning the projects and by studying modelling results. This feedback were used to further refine UIM.

4.2. Questionnaire

In one of the projects the method was evaluated using a questionnaire (Lif, 1997). In this project we did not participate. Here, UIM was performed by a group consisting of five end-users, one method leader, two software engineers responsible for the interface design, and one supervisor. All members of the modelling group were given a questionnaire to answer. The questions were divided into two parts. The first part measured the ease-of-use of the method and the second part measured how useful the result of the modelling was as input to the design process. A four-point scale was used for each question, where four indicated the highest rating. The mean values for ease-of-use and for practical utility were 2.94 (std.dev = 0.60) and 4.00 (std.dev = 0), respectively.

4.3. User study

During a half-day tutorial in UIM at RSV a user study was performed where different groups of users used the method. This tutorial was a part of a 2-day course in UML. The purpose with the study was to test if the participants could create an acceptable model of a limited system within a given range of time (30 min/task). The 9 participants were all software engineers with some knowledge in object-oriented development. The subjects were first introduced to the method theoretically and were than given some practical tasks. They were divided into groups by three and they were all modelling the same system, a system in which they had great domain knowledge.

The results of the evaluation showed that all participants were able to create clear and distinct models within the given range of time. All participants got similar results in the tests.

4.4. Interviews

UIM is general to its nature so when using it in a project it is sometimes necessary to adjust it to suit the specific project. To get a deeper understanding in how the method has been applied in different projects and how useful it has been, three interviews were made. From two of the projects the modelling leader was interviewed. In a third project one of the software engineers, responsible for the design, who were deeply involved in the modelling process was interviewed. The different interviews were similar in terms of procedure and content and a checklist with important questions was used to guide each interview.

All interviews were recorded and written down. The result was carefully analysed and the information was categorised according to the six main questions in the checklist.

1. How was the method used?
   In one of the projects the method was used in conjunction with use cases. The other projects used routine sketches instead of use cases. A routine sketch is a sequential description of a user's task. In the modelling sessions' users, software engineers and a modelling leader participated. The user interfaces were designed by the software engineers. The different steps were performed more or less as described in the method. However, in one of the projects the characteristics of the attributes were not modelled together with the users. Instead the users described these characteristics in free text that was later "translated" to the proper form by the GUI-developer.

2. Difficulties with the method
   All interviewed thought that the most difficult task was to identify the work situations on a suitable level. In one of the projects the participants realised that the work situations had become too "small" which led to a new modelling session where some of the work situations were merged together.

   Some of the users had problems understanding how to describe the characteristics of the different attributes. Here, the users were not led by a modelling leader.

The interviewed thought it would have been easier to perform this task in modelling sessions, as described in the method.

3. The usefulness of the method
   The result of the UIM was partly used as an input to start the design process and later on to check if the content of the interface was correct. However, the models specified in UIM were not enough. It was also necessary to have a dialogue with the user group during the whole design process to get feedback on different design solutions.

4. Ease of learning
   The subjects had different opinions on how easy it was to learn to use the method. In one of the projects they did not have any difficulties in using the method. In another project the participants spent a lot of time discussing the work situation concept. One of the interviewed thought it was difficult to explain what a work situation is, but when that was clear it was easy to proceed.

5. Comments from the subjects on the latest version of the method
   The latest version of the method (as described in this paper) was presented and discussed. All agreed that this version seemed to be more distinct and that the results seemed to be useful. They also agreed upon that when using the older version there was a risk that bad design solutions could occur due to confusing documentation of the information objects. They thought this version was more obvious.

6. Possible improvements
   One opinion was that it is important to inform the participants of the purpose of specifying the different models. Another opinion was that too little time was spent on UIM in the project. More time should have been reserved for deriving a stable model. The importance of including a designer early in the development process was also stressed.

5. Discussion

The method has shown to be useful in several projects at the Swedish National Tax board. The questionnaires and the user study performed during the workshop are not extensive evaluations but give at least a hint on its usability. Together with the experiences from different applied projects and the interviews, the overall impression is that it is both easy to use and a useful input for the user interface design.

There are of course some difficulties when using this method. It is not always obvious how “big” a work situation should be, i.e. which use cases to include in a work situation. It differs from project to project. The same problem occurs when modelling the use cases. The size of a use case and the size of a work situation depends on the problem domain. Experience is needed to get a feel for this. Another problem with this method is to be abstract enough to not limit the design space, and be concrete enough to make it possible to understand for all participants. The modelling leader is the key person here. He has to be able to keep the model an a level of abstraction that all participants can comprehend without making decisions concerning the style of the interface.

Modelling the users work is always time consuming. Using this method is not an exception. The pay back comes later on in the development process. The outcome of this modelling are requirements that are useful when making the design of the user interface. If the information is not available the designer has to get it some other way and that takes a considerable amount of time. If the designer does not get the information there is a great risk that the information system will be both ineffective and unpleasant to use. This will either lead to demands on new and better design solutions or even worse, slow down the users work and give rise to mental work load. Therefore it is necessary to model the requirements on the user interface.

5.1. User centred design

User centred design has become rather common in in-house development projects (Grudin, 1991). Involving the users in the development project is undoubtedly important. The users are the only experts on the work to be supported by the system and they should therefore have a great influence on the result. However, when working in a user centred development project it is important to carefully consider how to involve the users in the most beneficial way. The users are experts on their work, not on designing interfaces. The analysis of the users' work should only describe the contents of the users' work, not the style of the interface.

5.2. Including a designer

Designing a user interface is a complex process. Knowledge in software engineering, cognitive psychology, usability and some artistic capabilities is needed. Too often the user interface is just regarded as the cosmetics of the product. Therefore too little time and effort are spent on interface design. Involving a designer with responsibility for the user interface, would help bridging the gap between analysis and design. The designer can act as a link between the users and the software engineers (Figure 6).

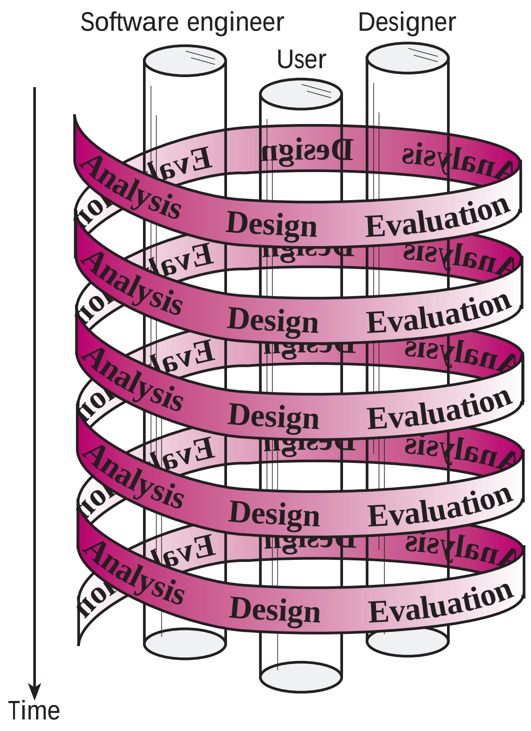

Figure 6. A designer communicates with software engineers and users during the iterative development process.

In the future we will strive to further narrow the gap between analysis and design. User interface design is partially a creative process. By supplying the designer with relevant information more time can be spent on the creative work of making design decisions and less time on studying irrelevant modelling results.

References

BENYON, D. (1992). The role of task analysis in systems design. *Interacting with computers, 4* (1), 102-123.

BOOCH, G., JACOBSON, I., & RUMBAUGH, J. (1997). *Version 1.0 of the Unified Modelling Language,* (On-line). Available: http://www.rational.com/uml/references/docset.html.

CARD, S.K., MORAN, T.P., & NEWELL, A. (1983). *The Psychology of Human-Computer Interaction.* Hove, England: Lawrence Erlbaum Associates, Inc.

CONSTANTINE, L. (1995). Essential Modelling: Use Cases for User Interfaces, *interactions, 2*, 34-46.

DEMARCO, T. (1978). *Structured Analysis and System Specification.* New York: Yourdon Press.

FLOYD, C. (1986). A comparative evaluation of system development methods. In T.W. Olle, H.G. Sol, & A.A. Verrijn-Stuart (Eds.), *Information systems design methodologies: Improving the Practice* (pp.19-55). Elsevier Science Publishers B.V.

GOULD, J.D., BOIES, S.J., & LEWIS, C. (1991). Making usable, useful, productivity. Enhancing computer applications. *Communications of the ACM, 34* (1), 74-85.

GREENBAUM, J., & KYNG, M. (Eds.). (1991). *Design at Work: Cooperative Design of Computer Systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

GRUDIN, J. (1991). Interactive Systems: Bridging the Gaps Between Developers and Users. *IEEE Computer, 24* (4), 59-69.

GULLIKSEN, J., LIF, M., LIND, M., NYGREN, E., & SANDBLAD, B. (1997). Analysis of Information Utilisation. *International Journal of Human-Computer Interaction, 9* (3), 255-282.

GULLIKSEN, J., LIND, M., LIF, M., & SANDBLAD, B. (1995). Efficient Development of Organizations and Information Technology – A Design Approach. In Y. Anzai, & K. Ogawa (Eds.), *Symbiosis of Human and Artifact. Proceedings of the 6th International Conference on Human-Computer Interaction, HCI International' 95* (pp. 951-956). Amsterdam: Elsevier Science B.V.

ISO/DIS 9241-11 (Draft). Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability (1995). Geneva, Switzerland: International Organization for Standardization.

JACOBSON, I., CHRISTERSON, M., JONSSON, P., & ÖVERGAARD, G. (1992). *Object-Oriented Software Engineering. A Use Case Driven Approach.* Wokingham, England: Addison-Wesley Publishing Company.

LIF, M. (1997). User interface modeling for design of administrative information systems. In *Proceedings of the 7th International Conference on Human-Computer Interaction, HCI International' 97* (pp. 383-386). San Francisco, U.S.A: Elsevier.

MACLEAN, A., YOUNG, R.M., BELLOTTI, V.M.E., & MORAN, T.P. (1991). Questions, Options and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction, 6*, 201-250.

MULLER, M.J., HASLWANTER, J.H, & DAYTON, T. (1997). Participatory Practices in the Software Lifecycle. In M. Helander, T.K. Landauer, & P. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (pp. 255-297). Amsterdam: Elsevier Science B.V.

PREECE, J., ROGERS, Y., SHARP, H., BENYON, D., HOLLAND, S., & CAREY, T. (1994). *Human-Computer Interaction.* Wokingham, England: Addison-Wesley Publishing Company.

RAPOPORT, R. (1970). Three dilemmas in action research. *Human Relations, 23* (6), 499-513.

SUTCLIFF, A.G., & WANG, I. (1991). Integrating Human Computer Interaction with Jackson System Development, *The computer journal, 34*, 132-142.