Journal of Universal Computer Science, vol. 9, no. 1 (2003), 51-85
submitted: 23/1/02, accepted: 24/1/03, appeared: 28/1/03 © *J.UCS*

Experimental Studies within the Software Engineering Process for Intelligent Assistance in a GUI

Maria Virvou
(Department of Informatics, University of Piraeus,
80 Karaoli & Dimitriou St., 18534, Piraeus, Greece
mvirvou@unipi.gr)

Katerina Kabassi
(Department of Informatics, University of Piraeus,
80 Karaoli & Dimitriou St., 18534, Piraeus, Greece
kkabassi@unipi.gr)

Abstract: This paper presents the research work towards improving human computer interaction by providing intelligent assistance to users. This has been approached by incorporating principles of a cognitive theory in a Graphical User Interface (GUI), that deals with file manipulation and is called IFM. The cognitive theory is called Human Plausible Reasoning (HPR) and has been used to simulate users' reasoning in the user model of the system so that the GUI may provide spontaneous assistance to users' errors. Such a goal is difficult to achieve and depends heavily on the development process. However, there is a shortage of reports on the software engineering process of intelligent assistants. Moreover, in the literature of intelligent assistants there is evidence that some important phases of their development process may have been omitted and thus the understanding of delicate issues has not improved significantly. Therefore, the focus of this paper is on presenting and discussing the software engineering process of the intelligent assistant developed. Special emphasis has been put on the description of the experimental studies, which were conducted prior and after the development of the system. Theses studies were used for the specification and refinement of the overall design as well as the adaptation of HPR in it. The experimental results have shown that the intelligent assistant may follow the users' reasoning and provide helpful advice to a satisfactory extent as compared to human advisors.

Keywords: Intelligent Help, intelligent user interface, object-oriented software engineering, user modelling, experimental studies.
Category: D.2.10

1 Introduction

As the number of software users increases dramatically throughout the world, the need for improving Human-Computer Interaction becomes more apparent and demanding. This is especially the case for software, which is addressed to a wide range of users of various backgrounds, ages, levels of skills, preferences and habits. Software of this kind includes programs for file manipulation such as the Windows 98/NT Explorer. Programs like this are used by anyone who wishes to download files from the Internet, create new files from word processors, spreadsheets and other packages, create their own programs, copy and move files from one disk to another

etc. Obviously, tasks of this kind are carried out by the largest portion of users ranging from experienced users to novice ones. Novice users may encounter difficulties due to lack of experience and expert users may face problems due to carelessness or possible tiredness. However, traditional on-line help is not always sufficiently helpful. For example, Matthews et al. [Matthews et al., 00] highlight the fact that on-line manuals must explain everything and novices find them confusing, while more experienced users find it quite annoying to have to browse through a lot of irrelevant material. Therefore, in addition to basic user interface guidelines, developers may also consider the benefits of using knowledge-based methods to enhance user interfaces.

Indeed, a lot of research energy has been put into the development of intelligent user interfaces. Quite a lot of them focus on providing intelligent help to users who encounter problems during their interaction with the computer (e.g. [Wilensky et al., 00], [Matthews et al., 00], [Jerrams-Smith, 00], [Horvitz et al., 98]). Most such systems incorporate a user modelling component which is responsible for understanding the person that interacts with the system. The user model accounts for user behaviour which is the observable response to a particular stimulus in a given domain [Sison & Simura, 1998]. In this sense, the user modelling component takes as input observable actions or queries/answers of users and tries to infer the real users' intentions, beliefs, level of knowledge, possible misconceptions etc. In intelligent assistance, these inferences about the user are used by the system to generate automatic assistance adapted to the users' needs.

In the context of intelligent assistance, we have developed a knowledge-based GUI that intends to offer spontaneous help to users who have problems due to their own mistakes. In particular, the GUI aims at understanding the users' reasoning when they make plausible human mistakes in their effort to conform with the interface's formalities and achieve their goals. The domain that we selected to examine the capabilities of the intelligent assistance is one that is addressed to a wide range of users. Therefore, we developed a GUI that manages files and folders in a similar way as the Windows 98/NT Explorer [Microsoft Corporation, 98] which is a program used by a very large portion of computer users with varying backgrounds and levels of skills.

The knowledge-based GUI is called Intelligent File Manipulator (IFM) and deals with the management of files and folders. IFM monitors users' actions and reasons about them in terms of users' intentions and possible mistakes. This kind of reasoning is performed by the user modelling component of IFM. In case IFM judges that the user may have made a mistake with respect to his/her hypothesised intentions, it suggests an alternative action that the user may have really meant to issue rather than the one issued. Therefore, the reasoning of IFM may also be used in a learning environment for novice users since it protects them from erroneous, destructive actions [Virvou and Kabassi, 02].

User modelling in IFM is largely based on an adaptation of a cognitive theory, called Human Plausible Reasoning theory [Collins and Michalski, 89], henceforth referred to as HPR. HPR is a domain-independent theory originally based on a corpus of people's answers to everyday questions. Starting from a question asked to a person, the theory tries to model the reasoning that this person employs in order to find a plausible answer, assuming that s/he does not have a ready answer. In IFM, we have

used this theory to simulate the analogical reasoning of users that may lead them to plausible human mistakes during their interaction with the system.

However, the actual development process of an intelligent user interface, like IFM, is an issue that needs a lot of attention. Höök [Höök, 00] points out that there are a number of problems not yet solved that prevent us from creating good intelligent user interface applications; one of these problems is that we do not have efficient methods for developing such applications and that we need a better understanding of the possible ways the interface can utilise intelligence to improve the interaction. The shortage of guidelines available for the development of intelligent user interface applications is also highlighted by other researchers as well. For example, Delisle and Moulin come to this conclusion in their review of the literature of intelligent help systems [Delisle and Moulin, 02].

In view of the above, in this paper we present and discuss the development process of IFM throughout its life-cycle. Special emphasis has been put on presenting the experimental studies that were conducted and used for the requirements analysis, design specifications and empirical evaluation of the user model and the system. In particular, the experimental study prior the development of IFM was used for clarifying the kind of intelligent assistance that IFM was going to provide. Moreover, it was also used for refining and adapting HPR theory in the particular context of intelligent assistance in a GUI. The experimental studies that were conducted for the evaluation of IFM ensured that the system addressed real users' needs and provided helpful assistance to a satisfactory extent. The life-cycle model of IFM has been based on the Rational Unified Process [Kruchten, 99], [Quatrani, 98] which advocates multiple iterations of the development process.

The main body of this paper is organised as follows. In Section 2 we present and discuss related work in intelligent assistance. In Section 3 we describe the experiment that was used for requirements analysis. In Section 4 we give a brief description of IFM's current design and its operation. In Sections 5 and 6 we present our approach in evaluating IFM. Finally, in Section 7 we discuss this work and give the conclusions drawn.

2 Related Work

The research goal of IFM for automatic generation of intelligent assistance is shared by a lot of other systems in the area of Intelligent Help Systems (IHSs) [Delisle and Moulin, 02]. However, the approaches to the kind of assistance as well as the way that this assistance is generated vary considerably.

In terms of the kind of assistance provided, there are IHSs that respond to explicit users' requests such as UC [Mayfied, 92], [Chin, 89], [Wilensky et al., 00] and AQUA [Quilici, 89], which are IHSs for UNIX users. Such systems may be of help to users who have realised that they need assistance. However, in day-to-day interaction of users with software applications there are many cases when users involve themselves in problematic situations without their realising it. These problems may be addressed by systems that monitor users silently and respond spontaneously to problematic situations, such as the Office Assistant [Horvitz et al., 98] that provides spontaneous help to users working with Microsoft Office and Microsoft's Tip Wizard, which is a similar system to the Office Assistant.

However, even in cases of spontaneous interventions there are different approaches concerning what is considered problematic. For example, the Office Assistant mainly intends to help users by optimising their plans, which may be correct rather than help them with their errors. The same goal is also shared by USCSH [Matthews et al., 00], which is a help system for UNIX users. USCSH aims at showing users better and more efficient ways of getting a task done. Unlike these systems, IFM aims primarily at helping users in situations where they accidentally issue actions, which they do not really intend. Such actions include commands that are prompted with error messages by a standard explorer. However, most importantly, they also include actions, which may be syntactically correct with respect to a standard explorer's formalities but they do not achieve what the user may have really meant. For example, a user may accidentally delete a file, which was useful.

More specifically, IFM aims at reproducing the human reasoning of a colleague or an expert sitting next to a user and observing his/her actions. This is the reason why IFM's reasoning is primarily based on a cognitive theory of human plausible reasoning. This reasoning is used to imitate a user's reasoning, which may be correct or incorrect but in any case plausible. In this sense, IFM incorporates a unified framework for handling both correct and incorrect user actions. Similarly to IFM's approach, Eller and Carberry [Eller and Carberry, 92] describe a set of meta-rules for hypothesising the cause of errors in ill-formed dialogues. Their domain is not interactive software but naturally occurring dialogues. In their context, the system performs a relaxation of the semantic interpretation of a user's utterance that allows the interpretation of the utterance in a less precise way than it was originally perceived. This relaxation occurs in situations when the system has problems in assimilating the user's plans and goals. In IFM, a similar form of relaxation takes place when an erroneous action is transformed through the use of HPR by the system in its effort to gain an understanding of what the user's real intention was. In the case of IFM, HPR provides the advantage of a relatively domain-independent method of relaxation.

Indeed, HPR has been previously used in another IHS of a different domain. That system was called RESCUER [Virvou, 99], [Virvou and du Boulay, 99] and provided automatic assistance to UNIX users. The user interface of UNIX is a command language interface, which is different from a graphical user interface that involves mouse events. Moreover, command language interfaces are considered less user-friendly than GUIs and are probably used by a smaller number of computer users than GUIs. Therefore, the exploration of the utility and application of HPR in a GUI after it has been applied in a command language interface is very useful. In particular it

reveals the potential of HPR for a more general framework for the development and incorporation of intelligent human-like help into user interfaces.

However, one very important issue that seems to have been overlooked in the literature of IHSs is the actual development process of such systems and the software engineering techniques that should ideally include experimental studies in several phases of the software life-cycle. Such phases definitely include the requirements analysis as well as the evaluation of IHSs. For example, Chin [Chin, 01] points out that empirical evaluations are needed to determine which users are helped or hindered by user-adapted interaction in user modelling systems. He adds that the key to good empirical evaluation is the proper design and execution of the experiments so that the particular factors to be tested can be easily separated from other confounding factors. However, he notes that empirical evaluations are not so common in the user modelling literature. Similarly, Mc Tear [McTear, 00] points out that the relationship between theory and practice is particularly important in Intelligent Interface Technology as the ultimate proof of concept here is that the interface actually works and that it is acceptable to users; for this reason practical issues such as performance, reliability and usability would seem to be more important than theoretical issues such as choice of system design methodology or specification notations.

In view of these, IFM has been developed based on the Rational Unified Process [Kruchten, 99], [Quatrani, 98], which is an object-oriented model that advocates multiple iterations in the software life-cycle. This iterative software life-cycle involved crucial experimental studies. One experimental study was conducted for requirements analysis and another one was conducted for the empirical evaluation of the system and its user modelling capabilities. Both studies involved both users (novice and expert) and human experts that acted as silent observers of the users' actions.

One important advantage of these studies was that for each protocol of a particular user's actions there were many human experts who acted as observers and were asked to comment on these user's actions. Each human expert gave his/her comments independently of the others. This gave us insight on what is possible to be modelled and what kind of help may be given. In particular, there were cases where there was a high degree of diversity of opinions among human experts. In such cases we considered it impossible for an IHS to be able to provide human-like advice with a high degree of certainty since not even real humans could provide such advice. In contrast, there were many cases where experts had a unanimous opinion about what the user was doing and the kind of help s/he needed. The provision of automatic help in problematic situations of this kind was considered among the necessary functional requirements of the system. Therefore, in the evaluation of the system we also conducted experiments where many human experts could express an opinion for the same user's actions. Then these opinions were compared among them and then with IFM's responses.

Other experimental studies that have been conducted within the development of IHSs tend to rely on the opinion expressed by one human expert for each protocol. For example, in the Lumiere Project [Horvitz et al., 98] and an IHS for UNIX users [Jerrams-Smith, 00], there are reports on experimental studies where there was only one human expert per user protocol. Therefore, there could not be a comparison among different human experts' opinions concerning the same user actions. However,

even though the approach of these systems in the experiments was different from IFM's there was an acknowledgment of the fact that human experts were typically uncertain about a user's goals and the kind of assistance needed. This reinforces the view that special effort must be made so that the requirements analysis and evaluation may throw light on what is really needed and what is possible to be achieved in IHSs.

3 Experimental Study for Requirements Analysis

IFM's life-cycle was based on Rational Unified Process, which divides the development cycle in four consecutive phases: the inception, the elaboration, the construction and the transition phase. In the inception phase the primary executable release of IFM was developed. IFM is a system that constantly reasons about users' actions in order to diagnose problematic situations and give advice concerning the error identified. Examples of erroneous actions include clicking on the wrong command or even the wrong files.

3.1 The Experiment

At the early stages of IFM, we conducted a usability evaluation of a standard file manipulation program, which does not incorporate intelligence, such as Windows 98/NT Explorer. This evaluation aimed at identifying usability problems of standard file manipulation programs so that these problems were addressed in the design of the next version of IFM. For this reason we conducted an experiment, which involved both users and human advisors.

One of the main aims of the empirical study was to categorise as many users' plans as possible and to identify the most frequent errors that expert and novice users may make while interacting with a standard explorer. In this way, we could identify limitations of IFM. Another important aim of the empirical study was to evaluate IFM's reactions, in comparison to the human expert comments. The results of that comparison were also used in order to identify IFM's limitations so that we could enhance the specifications for a second version of IFM.

The experiment involved 30 novice and expert users of a standard explorer. All users were asked to use a standard explorer as they would normally do in their day-to-day activity. Moreover, there were 10 human experts acting as potential advisors of users. All human experts possessed a first and/or higher degree in Computer Science and had teaching experience related to the use of such programs.

The experiment consisted of 3 phases, in a similar way as the 3 initial phases of the empirical study described in [Sutcliffe, Ennis and Hu, 00]. In particular there was a pre-test questionnaire and interview, then there was a short system training for novice users and then the main experimental task.

The main experimental task consisted of usability tests concerning a standard explorer. Users worked on a standard explorer and their actions were recorded. Then each protocol of users' actions was given to all of the human advisors to be analysed. These advisors were asked to identify users' mistakes (with respect to the users' hypothesised intentions) in the protocols. Advisors were also asked to write down what they thought the most appropriate piece of advice would be for the mistakes that they identified. The comments of the human advisors in each protocol were compared among them. In cases where the majority of human advisors identified the same

mistake, it was considered that indeed the user had made a mistake. Then the advisors' comments for remedy were compared to the standard explorer's responses or absence of response. In total, the human advisors examined 1342 actions of all users. Among those actions 753 were issued by novice users and 589 were issued by by expert users as can be seen in Table 1 that summarises the results of the study.

In the novice users' protocols, there were 154 actions that were possibly unintended according to the majority of human advisors. This corresponded to 20% of the total actions issued by the novice users. In addition, in the protocols of expert users, the majority of human experts identified 38 possibly unintended actions, which were mainly issued due to carelessness. The standard explorer, on the other hand, identified 112 possible errors, which accounted for 14.8% of the total actions of the novice users and 29 possible errors in the expert users protocols. However, even in cases when the standard explorer identified user errors, these errors were not necessarily the same as the errors identified by human experts. Moreover, the advice that the standard explorer provided was not always adequate. Consequently, 47 users' errors resulted in the loss of files that were valuable for the users.

Actions	Novice Users	Expert Users	Sum
Total actions	753	589	1342
Total possibly unintended actions according to the standard explorer	112 (112/753= 14.8%)	29 (29/589=4.9%)	141 (141/1342= 10.5%)
Total possibly unintended actions according to the majority of human experts	154 (154/753= 20%)	38 (38/589= 6.4%)	192 (192/1342= 14.3%)
Total possibly unintended actions with destructive result according to the majority of human experts	36 (36/753= 4.8%)	11 (11/589=1.9%)	47 (47/1342=3 .5%)

Table 1: Summary of the analysis of users' protocols

3.2 Usability Problems

The errors that were identified by the human advisors were caused due to several reasons that constituted usability problems of standard file manipulation programs. Some examples of problems that users of a standard explorer often encountered were the following:

The structure of a regular explorer was the source of many errors for most users and especially to novices. For example, users often tangled up the parent folder at the left part of the explorer with the folder shown at the right part of the program.

Another cause of confusion, especially for novice users, involved commands that were executed at two stages. This was also very confusing to users with previous experience in command language interfaces. For example, copying or moving an object from a directory to another was executed using one action only in command-language interfaces but needed two actions to be completed in graphical user interfaces. So users often copied or cut one or more files but did not complete the copy or move operation, respectively.

A serious error category, committed by both novice and expert users, concerned the deletion of objects; users often deleted a folder without being aware of its content. This error category was dangerous because the results could be devastating. In a standard explorer, some measures have been taken against the loss of useful information due to accidental deletion of files or folders. For example, there is a confirmation message after the execution of the deletion command and there is also the recycle bin. However, the confirmation message gives information about the name of the folder deleted, only after the user has deleted one folder.

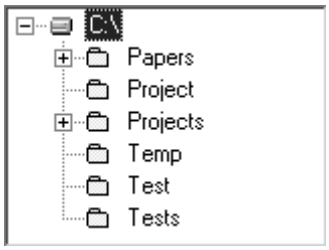

Figure 1: The user's initial file store state in the first example

For example, an error that was made by one user in the protocols examined concerned the deletion of four folders (the user's file store state is shown in Figure 1). The user selected the folders 'Projects', 'Temp', 'Test' and 'Tests' to delete them. However, the user did not really intend to delete the folder 'Projects' but the folder 'Project'. The cause of the error was due to the fact that the user had accidentally selected the folder called 'Projects' instead of the folder called 'Project', which was empty. The standard explorer produced a confirmation message about the deletion of the four folders (Figure 2), but it did not mention the names of the folders to be deleted. Furthermore, it had no reasoning mechanisms to recognise that the deletion of the folder 'Projects' was probably undesired by the user. Therefore, the user deleted the four folders without realising s/he had made an error.

Figure 2: The confirmation message of the standard explorer for the deletion of 4 items.

The consequences of the above mentioned error could have been more destructive if the user had skipped the Recycle bin before realising that s/he had deleted the wrong folder. Indeed, expert users usually omitted intermediate stages in deletion plans. In the protocols selected there were several cases when the expert users omitted the Recycle bin by pressing the shift key. For example, an expert user wanted to delete the folders 'bill1' and 'games' (Figure 3). He selected the folders "bill2" and "games", he pressed the shift key and then selected the command delete.

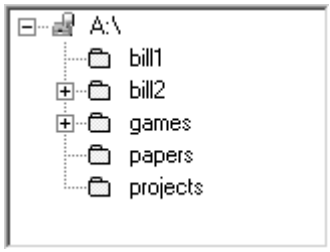

Figure 3: The user's file store state in the second example

Being mistakenly sure of his selection, he responded positively to the warning message of the standard explorer: "Are you sure you want to delete these 2 items?" However, the user's real intention was to delete the folder "bill1" rather than "bill2". The warning message of the standard explorer informed the user that two items were going to be deleted but it did not say which items. Moreover, the empirical evaluation revealed that users, especially expert ones, usually responded to this message without reading it, since it is always the same. Therefore, the user deleted the folder "bill2" and lost valuable data.

Finally, another issue that seemed very confusing for all users and especially for novice ones was that the recycle bin does not exist in removable drives. This is evident in the sample session presented below.

An intermediate user deleted the folder 'News' from disk A. However, she then realised that she had been mistaken and she did not really mean to delete the folder 'News'. Therefore, she tried to regain the deleted folder. Not having realised that the folder had been permanently deleted, she thought that it ended up in the recycle bin.

Therefore, she tried to find it in the hard disk C:\ and she used the commands of the window presented in Figure 4. However, the result of her search of the string 'News' in C:\ resulted in the system finding 348 files/folders, none of which was the desired one, and the user was further frustrated.

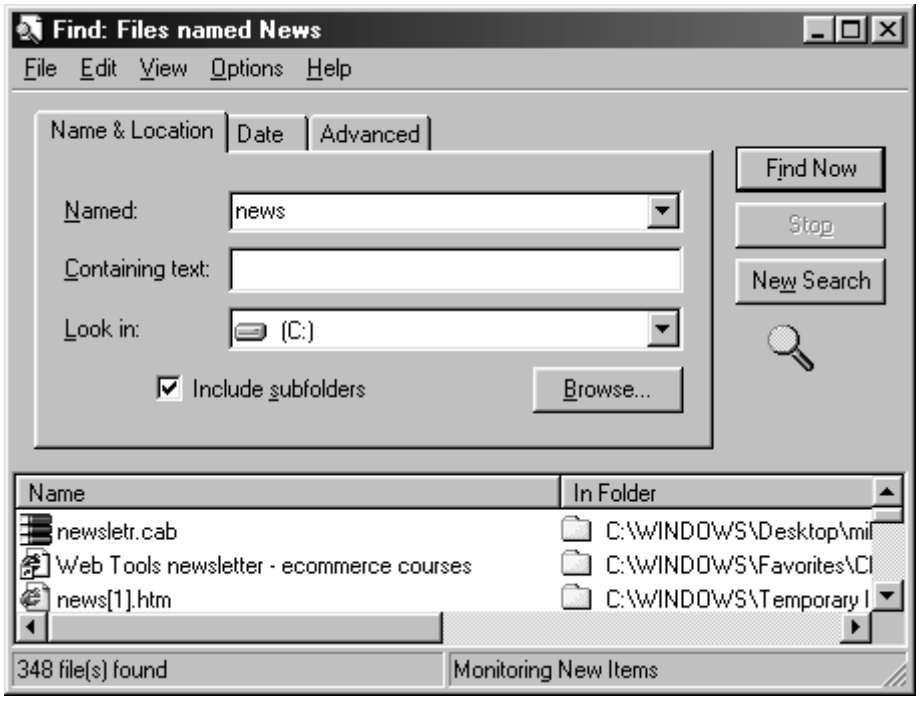

Figure 4: The user tries to find the lost folder

A subset of the previous error category concerned users that deleted an object after they had copied it, though they had not pasted it yet. For example, a novice user wanted to copy the file 'exams.doc' from disk A:\ to the folder C:\My Documents\. Therefore, he executed the command copy and then deleted the file. When the user selected the folder C:\My Documents\ and executed the command paste the system produced the error message in Figure 5. The user tries to find the file in the disk A:\ but unfortunately, it had been permanently deleted.

Figure 5: Error message for copying a file that has been deleted

The error probably occurred because the user believed that the object had been temporarily stored in the clipboard. However, the clipboard only stores the location of the object selected to be copied. In case the object is deleted and then pasted, the system is not capable of recovering the deleted object. In the particular case, the novice user's real intention was to move the file 'exams.doc' from a removable disk to his/her hard disk, but the user was not aware of the usage of command cut.

In addition, help provided by standard file manipulation programs did not seem to be sufficient, especially for novice users. Asking for help in standard file manipulation programs, presupposes that the users know how to ask for help [Virvou, Jones and Millington, 00]. This is one of the main reasons why users, especially novice ones, encounter many problems in the use of programs and make many errors that result in failure to achieve their goals. Moreover, both expert and novice users tended to reproduce the same kind of mistake over and over. Therefore, the help given to users should be more individualised by taking into account their history record. This could be achieved by keeping a long term user model for every user [Rich, 99].

As a result of the identified usability problems, the human experts categorised all users' errors in five categories. These categories are presented below:

- *Command errors*: Cases where the user had selected the wrong command with respect to his/her hypothesised intentions or cases where a command had failed. For example, some users confused the usage of 'cut'/'copy' and 'paste' commands.
- *Structure errors*: Cases where the user had made mistakes due to his/her unawareness of the structure of a standard file manipulation program. For example, when the user confused the parent folder on the left part of the explorer with the folder shown on the right part of the program. These errors were mainly made by novice users due to their lack of knowledge about the system and its operations.
- *Spelling errors*: Errors that were made because a user tangled up objects with similar names.
- *Mouse errors*: Errors that were made because the user had tangled up neighbouring objects in the graphical representation.
- *Identical name errors*: Cases where the user confused objects with exactly the same name that were situated in different places in the file store.

In general, all errors belonging to the last three categories were considered as accidental slips, which means that the user tangled up neighbouring objects or commands in the graphical representation or objects with similar names, for example "Doc" and "Docs".

3.3 Use Cases

The users' plans recognised during the experimental study served as the basis for the construction of use case diagrams. Use case is a modelling technique incorporated in UML which may be used to describe what a new system should do or what an existing system already does [Eriksson and Penker, 98].

However, Muller, Haslewanter & Dayton [Muller, Haslwanter and Dayton, 97] point out that there may be 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 has been made up by software engineers to represent user actions. To overcome these problems it is necessary to model the use cases in participation with the end-users [Lif, 98]. Indeed, in the context of IFM, we constructed use case diagrams after having analysed the results of the experimental study where users and human experts had participated. In particular, we identified the use cases where users tended to have problems.

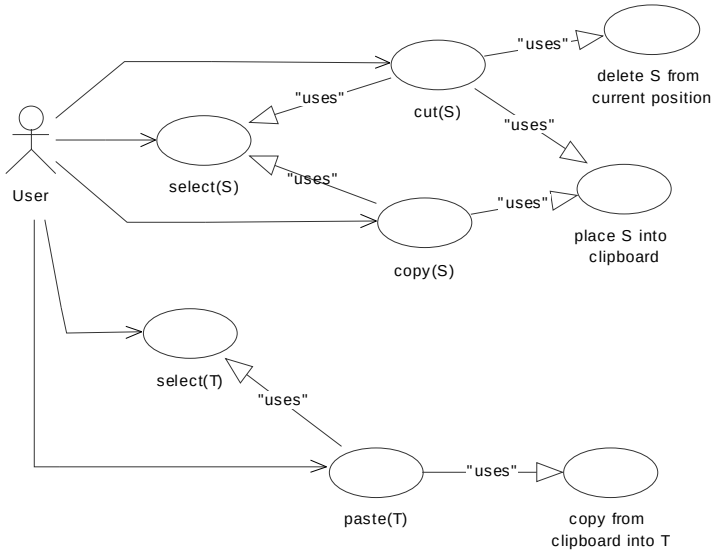

Figure 6: Use case diagram for “cut”, “copy” and “paste”

The use cases that have the relation communicates with the user, correspond to users’ actions. A «uses» relationship between use cases means that an instance of the source use case also includes the behaviour described by the target use case [Muller, 97]. In the context of our study, the «uses» relationship between use cases reveals similarities between user actions that may be confusing for users. For example, the use case diagram in Figure 6, illustrates the actions “cut”, “copy” and “paste”. In this diagram, it is shown that all three actions (cut, copy and paste) are related to a use case “select” which means that a user has to select an object before using these actions. Moreover, the actions “cut” and “copy” are related with a use case that places an object into the clipboard.

However, unlike the use case “copy”, in the use case “cut” there is one more relationship «uses» with the use case “delete the selected object from current position”. Therefore, the diagram of the example shows that “cut” and “copy” have a strong similarity in their functionality from the point of view of a user. The action “paste” bears also a similarity to “cut” and “copy”. Similarities like this have been taken into consideration for the construction of the underlying hierarchies of the system’s knowledge base as will be described in the next section.

4 Design and Implementation of IFM

In this section we describe the design of the latest executable release of IFM in the iterations of its life cycle.

4.1 Overall Description

Intelligent File Manipulator (IFM) is a graphical user interface for file manipulation that provides intelligent help to its users. IFM monitors users' actions and reasons about them. In case it diagnoses a problematic situation, it provides spontaneous advice. When IFM generates advice, it suggests to the user a command, other than the one issued, which was problematic. In this respect, IFM tries to find out what the error of the user has been and what his/her real intention was. Therefore, an important aim of IFM's reasoning is error diagnosis.

The reasoning of IFM is largely performed by its user modelling component which tries to model the user in terms of his/her possible intentions and possible mistakes. For this reason the user modelling component employs two reasoning mechanisms, which work independently of each other and are combined in order to show what the user may have really thought when s/he issued a command.

The first reasoning mechanism is used for recognising the users' goals and is based on what we call "instabilities". The second reasoning mechanism is based on HPR and is used to simulate the users' incorrect thinking that may have led them to possible mistakes. The second reasoning mechanism is also influenced by the recorded style, habits and error proneness of each individual user via HPR's certainty parameters as will be explained in more detail in Section 4.2.2. In this way, IFM may respond in a more adaptive way than a standard explorer since it can automatically adapt the messages that it generates to the individual user's needs.

IFM evaluates each user's action with respect to its relevance to the user's hypothesised goals. As a result of this evaluation each action is categorised in one of four categories, namely, expected, neutral, suspect and erroneous. Depending on the category, where it is categorised, the action is processed further by IFM or not.

In particular, if an action is compatible with the system's hypotheses about the user's intentions, it is categorised as expected. If it is neither expected nor contradictory to the user's hypothesised goals, it is categorised as neutral. In the cases of expected and neutral, the system executes the user's action normally without further notice.

If an action contradicts the system's hypotheses about the user's intentions it is categorised as suspect. Finally, if an action is wrong with respect to the user interface formalities it is categorised as erroneous. In the cases of suspect and erroneous, the system tries to generate an action other than the one issued that would fit better in the context of the user's hypothesised intentions. The alternative action has to be similar to the one issued and is generated based on HPR's statement transforms which transform the given action.

The categorisation of user actions in one of the four categories is done by the goal recognition mechanism which is based on instabilities. Instabilities are added and/or deleted from a list as a result of user actions. For example, when a user issues a cut or copy action this results in the addition of an instability to the list of instabilities. This instability is removed if the user issues a paste action. Another example is the creation of an empty folder, which also adds an instability because the system would

expect a subsequent user action by which the folder would acquire a content or be deleted. Indeed, the instability, which is associated with the existence of an empty folder, is deleted if a user issues an action that assigns some content to the empty folder or removes the folder. In this sense, an addition of an instability signifies the initiation of a user's plan whereas the deletion of an instability signifies the continuation of a plan.

An action is considered expected if it deletes at least one of the existing instabilities of the file store state. It is considered neutral if it neither adds nor deletes instabilities and suspect if it only adds instabilities although there are already other instabilities that have not been deleted or when an action violates an existing instability before this has been deleted as a result of another action. However, IFM uses the categorisation of user actions as a way of acquiring some idea about which action may need more attention. By no means does it intervene based only on the categorisation of commands.

A very simple example of an interaction of a user with IFM is the following:

The user created a new folder in the hard disk, inside the folder C:\My Documents\ which s/he called 'temp'. At that stage, the user's file store is shown in Figure 7. Then the user selected the file A:\unit1.txt and issued a cut command. This action was considered as neutral and was executed normally. However, the user then selected the folder C:\My Documents\temp and issued a copy command. IFM considered this action suspect because in case it was executed it would delete the content of the clipboard before it was used. In terms of instabilities, this action was suspect because it violated the existing instability that was associated with the "cut„ command before this was deleted as a result of another action. Therefore, IFM transformed the action based on the incorporated adaptation of HPR. The transformation of the given action is done in a way that similar alternatives which would not be suspect or erroneous can be found.

IFM evaluated the possible similar alternatives and found that the user probably meant to issue: paste(C:\My Documents\temp\) because 'copy' and 'paste' are very similar according to HPR transformations when applied to the hierarchy of users' actions (a part of which is illustrated in Figure 9). Moreover, the action paste(C:\My Documents\ temp\) is considered as expected because it uses the content of the clipboard and gives contents to the newly created folder 'C:\My Documents\temp\' which is empty. In this way it deletes two existing instabilities without adding any new one.

Therefore, the user is informed about the system's advice. However, the user is not obligated to follow this advice. S/he may execute his/her initial action or issue a completely new one. In the particular example, the user found the system's advice very helpful and, consequently, adopted its suggestion. Then, the user formatted the floppy disk, which was his/her final goal. In case the user had used a standard file manipulation program, his/her error in command 4 would not have been recognised and the user would have formatted the floppy disk and would have lost the useful file 'unit1.txt'.

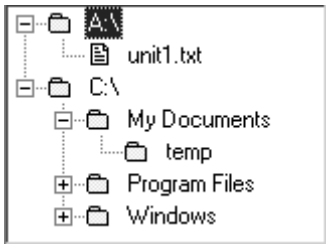

Figure 7: The user's file store state

4.2 The Knowledge Base of IFM

One important reasoning mechanism in IFM is based on an adaptation of HPR theory. HPR assumes that a person has a partial knowledge of a domain and when s/he is asked about something that s/he does not know, s/he tries to infer the answer from what s/he knows and is relevant to the question asked. The inferred answer may be correct or incorrect; in any case it is a plausible guess.

In IFM, we have used this reasoning to simulate users when they make "plausible" human mistakes. In our case, we assume that an error has resulted from reasoning that has been incorrect but is still plausible.

4.2.1 Hierarchies in Class Diagrams

In HPR, human knowledge about a domain is represented as a collection of statements. An example of a statement is: precipitation(Egypt) = very-light, which means that the precipitation of Egypt is very light. Precipitation is called a descriptor, Egypt is called an argument and very-light is called a referent. A descriptor is said to apply to an argument and together they form a term.

The simplest class of inference patterns are called statement transforms. Statement transforms exploit the 4 possible relations among arguments and among referents to yield 8 types of statement transform. There are eight statement transforms which allow plausible conclusions to be drawn. The *argument transforms* move up, down or sideways in the argument hierarchy using GEN, SPEC, SIM or DIS respectively. The *referent transforms* do the same in the referent hierarchy. For example, from the statement flower-type(England)=roses, we can make the following statement transforms, given the type hierarchy for geographic regions shown in Figure 8 and a similar type hierarchy for flowers (not illustrated).

Argument transforms

GEN flower-type(Europe)=roses

SPEC flower-type(Surrey)=roses

SIM flower-type(Holland)=roses

DIS flower-type(Brazil)≠roses

Referent transforms

GEN flower-type(England)=temperate flowers

SPEC flower-type(England)=yellow roses

SIM flower-type(England)=peonies

DIS flower-type(England)≠bougainvillea

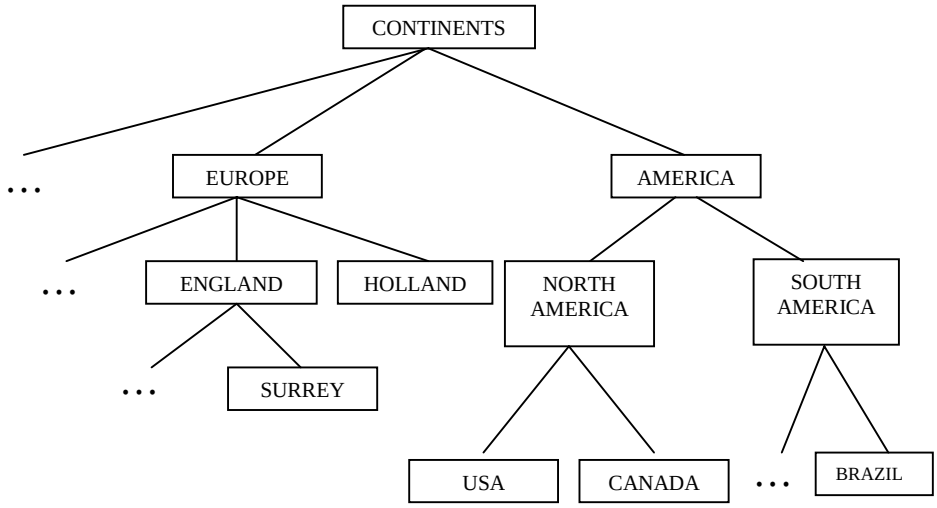

Figure 8: Hierarchy of geographic regions

The core theory also introduces certainty parameters, which are approximate numbers ranging between 0 and 1. Certainty parameters affect the certainty of different plausible inferences.

SIM and DIS statement transforms depend on the degree of similarity (σ), which represents the similarity of one set to another one. In particular, if the degree of similarity is almost 1 there is great confidence in the transformation, otherwise, the confidence decreases. The degree of typicality (τ) represents how typical a subset is within a set (for example, the cow is a typical mammal). Dominance (δ) indicates how dominant a subset is in a set (for example, elephants are not a large percentage of mammals). Finally the only parameter applicable to every expression is the certainty parameter (γ). This parameter indicates the degree of belief a person has that an expression is true. For example, in the formal representation of statement transforms the certainty parameter γ represents the degree of certainty of a person about this transform.

The domain knowledge in IFM concerns the use of commands and the representation of the file store state. Concepts concerning the use of commands are classified in hierarchies in order to be compatible with the main underlying

assumptions of HPR. However, the main hierarchy of IFM's knowledge base, that of users' actions represents procedural knowledge rather than declarative knowledge, which is used in example domains for HPR. In this respect, the use case diagrams that were specified during the requirements analysis provided an insightful basis for the construction of hierarchies out of procedural knowledge. This was mainly due to the nature of use cases which normally represent the functionalities of a system. Moreover, the relation «uses» among use cases highlighted shared functionalities, which were used for the classification of commands in hierarchies.

For example, the use case diagram in Figure 6 shows that the commands cut, copy and paste need an argument that has to be selected before using them. In addition, all three commands are related to the clipboard. Therefore, these three commands have been classified as commands that take an argument and have been placed under the "clipboard" commands; in this way they are in neighbouring positions in the hierarchy of commands. However, cut and copy share the same functionality concerning the placement of the selected object into the clipboard whereas the command paste does the opposite; it takes the content of the clipboard and places it into the selected object. Among the three commands, cut and copy are classified under the commands that "place into clipboard" and therefore they share a greater similarity. This can be seen in Figure 9, which illustrates the hierarchy of users' actions. The hierarchy represents the semantic and/or syntactic structure of actions. Moreover, it is constructed in such a way that every descendant node of a parent node inherits all the properties of the parent node.

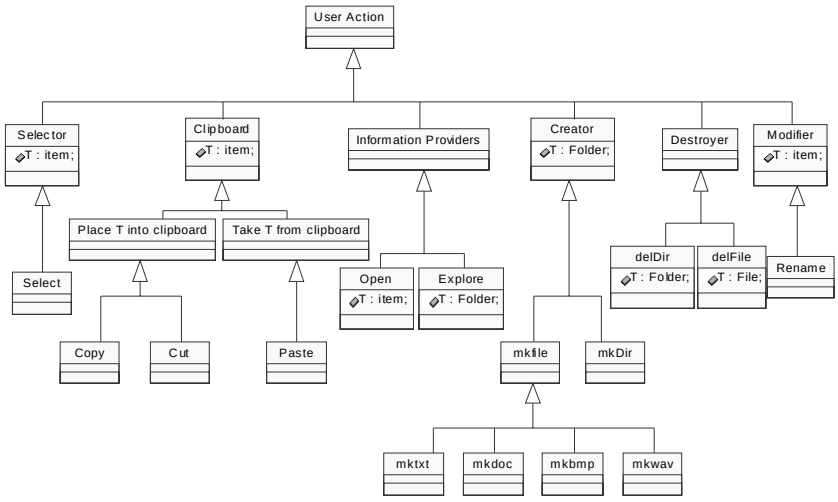

Figure 9: A Class diagram representing a part of IFM's knowledge base.

In general, we have found the UML notation quite suitable for modelling our knowledge base. In the case of Prolog, a term may be represented as a class and its arguments as attributes of the specific class. Furthermore, UML relationships can be used to represent predicates (also called *relations*). We can distinguish predicates into three categories according to their semantics. Predicates that correspond to an *isa*

relationship can be represented by a UML generalisation, for example isa(selector(T),userAction). Others that correspond to an ispart relationship can be represented by a UML aggregation, for example contains(folder,file). Finally, UML association represents predicates that cannot be categorised into one of the above categories, for example neighbouring(folder1,folder2). A hierarchical, tree-structured knowledge base may be presented by means of generalisations. In particular, in IFM we have used a class diagram in order to represent the structure of the Prolog predicates involved (e.g. Figure 9). Each term situated in a leaf of the tree structure corresponds to a user's action. Higher-level terms indicate the categorisation of users' actions, namely Selector, Clipboard, Information Providers, Creator, Destroyer and Modifier. The attribute T represents the item involved in user's action. T can be a folder, a file, or either a folder or a file (an item).

4.2.2 Generation of Advice

When IFM considers that an action issued by a user may have been incompatible with the user's hypothesised intentions, it tries to generate an alternative action that the user may have really meant to issue. The generation of possible actions to be suggested to the user is done based on HPR transforms. However, one possible problem in this approach is the generation of many alternatives.

A solution to this problem is ordering the alternative actions in a way that the ones, which are most likely to have been intended by the user, come first. The certainty parameters of HPR provide a good tool for ordering the alternatives. Certainty parameters are used in IFM in order to calculate a degree of certainty for every alternative action.

However, the certainty parameters of HPR were not immediately applicable in IFM. Their meaning needed to be specified in the domain of IFM. In addition, the exact way of calculation of one important certainty parameter, the degree of certainty (γ), was not specified fully in HPR. In view of these problems, the full specification of certainty parameters and their adaptation into IFM was done by taking into account the results of the experimental study that was conducted during the early phases of IFM's development.

We have used five of the certainty parameters presented in HPR: degree of certainty (γ), degree of typicality (τ) of an action in the set of all actions issued by the user, degree of similarity (σ) of a set to another set, frequency (ϕ) of an error in the set of all actions and dominance (δ) of an error in the set of all errors.

The degree of similarity is used to calculate the resemblance of two commands or two objects. The similarity between two commands of the hierarchy is pre-calculated. The value is estimated by taking into account the result of the commands, their relative distance in the user actions hierarchy and finally their relative geographical position in the graphical user interface. For example two commands that have a very similar result, such as "cut" and "copy" commands, have a great degree of similarity. Moreover, two commands that are neighbours in the user actions hierarchy, such as "mktxt" and "mkdoc", have a high degree of similarity. Finally, all users, and especially novices that have little experience, tend to entangle commands that are neighboring in the graphical user interface. In this case, the similarity between two objects is dynamically calculated. The value of the similarity of two objects is partially based on the resemblance of their names (e.g., directories "Doc" and "Docs")

but also based on relative distance of objects in the graphical representation of the file store.

The typicality of a command is based on the estimated frequency of execution of the command by the particular user. The degree of frequency of an error represents how often a specific error is made by a particular user. However, sometimes one has to know a particular user's weaknesses. Such weaknesses can be recognized by the dominance of an error in the set of all errors.

Finally, all parameters are combined in order to calculate a degree of certainty related to every alternative command generated by IFM. This degree of certainty represents the system's certainty that the user intended the alternative command generated. The degree of certainty determines whether this command is to be proposed to the user or not and if it is, in what priority.

The certainty of the system's advice is calculated as a sum of all certainty parameters, with each parameter being multiplied to a weight, which is determined with respect to how important the particular certainty parameter is. The formula of the degree of certainty is shown in equation (1).

$$\gamma = 0.4\sigma + 0.3\delta + 0.2\phi + 0.1\tau \qquad (1)$$

The weight of each certainty parameter was estimated based on the results of the experimental study that took place in the early stages of the development. The analysis of the comments of the human experts revealed important aspects that human experts took into account when they reasoned about users' actions in order to give advice. For an expert to suggest an alternative action, s/he had to evaluate candidate alternative actions in order to select the most appropriate one. The most important criterion when evaluating an alternative action, which was going to be proposed to the user, was the similarity of that action to the one issued by the user, because users generally tended to tangle up actions or objects that were very similar. Thus, in the formula, the weight of the degree of similarity is estimated to 0.4, which is the largest weight of all.

Human experts took seriously into account whether a particular error was the most common error of the user or not. So the weight of dominance of the particular error in the set of all errors is 0.3. An important criterion when evaluating an alternative action was the frequency a user made an error while interacting with the system. The degree of frequency of the particular error is multiplied by 0.2. Human experts were also interested to know if the user used the particular action that they were about to propose frequently or not. It was more likely that the user intended an action s/he has executed many times rather than another one s/he has never executed before. So the typicality of a certain command for the particular user also plays a role and is multiplied to 0.1.

5 Evaluating IFM's Reasoning in Comparison with Human Experts

After the construction of IFM was completed, the system was evaluated so that the usefulness of its operation could be ensured. One important aim of IFM was the development of a more adaptive and intelligent GUI than a standard explorer, which

would provide additional reasoning. This reasoning was aimed at rendering the interaction more human-like in terms of intelligent and plausible responses of the system to users' errors. Therefore, an important evaluation goal was to find out how successful IFM was at producing additional reasoning in comparison to a standard explorer. Moreover and most importantly, IFM was evaluated as to how successful it was at reproducing reasoning similar to human experts who observed the interaction.

For the above purposes, an experiment was conducted which was very similar to the one described in the requirements analysis. 30 novice and expert users were asked to interact with a standard explorer. Their actions were recorded and the protocols collected were given to 10 human experts who were asked to comment on them. This time, these protocols were also given as input to IFM and IFM's responses were also recorded. Then IFM's responses were compared to those of a standard explorer and to the comments that the human experts had made when they analysed the protocols. However, both the human advisors' comments and IFM's responses were not seen by the users who had interacted with a standard explorer. Hence, there were many cases where the correctness of IFM's hypotheses about the real intentions of the users could be verified by the users' subsequent actions.

5.1 A sample session

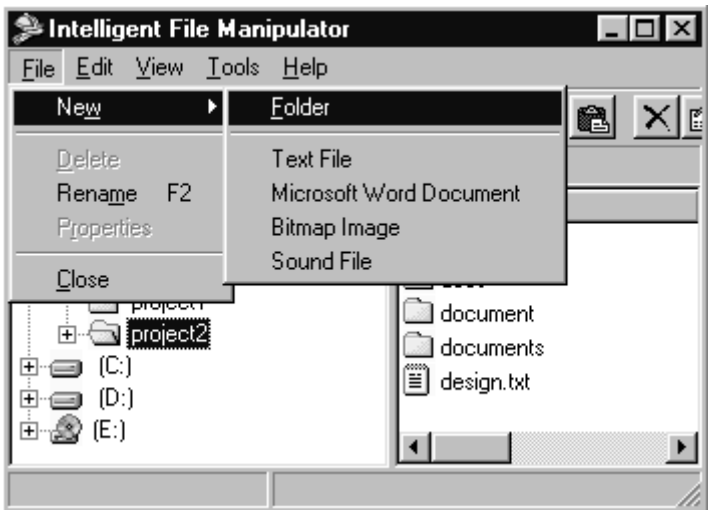

Figure 10: Actions for the creation of a new folder in A:\project3\

In Table 2, we illustrate a sample of a user protocol and show what IFM's reactions were to the user's actions and how these were compared to a standard explorer and to the reactions of human experts. The user's initial file store state is presented in Figure 11.

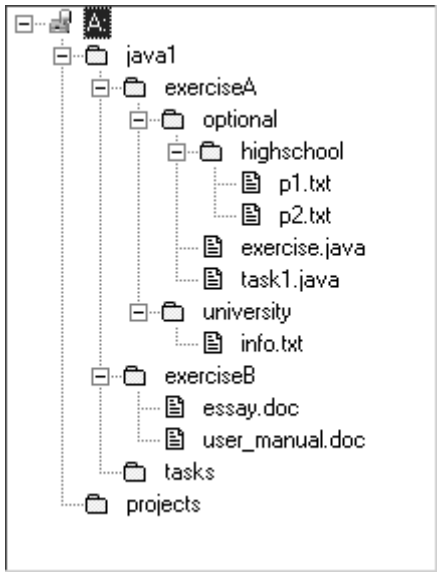

Figure 11: The user's initial file store state

In Table 2, there are four columns. The first column illustrates the actual user's commands. However, due to the limited space, we use the meaning of groups of actions rather than presenting the exact screenshots as these had been shown to human experts. For example, if a user had selected a folder (e.g. A:\project2\), then selected "File" from the menu bar, then selected "New", then "Folder" (as illustrated in Figure 10), this sequence of actions is represented by the command "create_new_folder_in(A:\project2\)", which is a synopsis of the meaning of these actions. The second column illustrates the reasoning of IFM that corresponded to each command; in case a command was characterised as suspect or erroneous, IFM generated alternative commands and suggested to the user to replace the command issued with one of the alternatives. The third column illustrates the responses of a standard explorer. Finally, in the fourth column we demonstrate whether IFM's suggestions were compatible to the human experts' suggestions for each command.

In the sample protocol in Table 2, the reader can see that IFM may follow the user's reasoning in a lot more cases than a standard explorer. Commands, which are considered "expected" by IFM show that IFM has a high degree of certainty that the user had intended these commands. In the sample protocol such commands are the commands 2, 5, 6, 12 and 14, which account for 1/3 of the total actions of the user.

Moreover, in the sample protocol, IFM was very successful at diagnosing two errors of the user at commands 8 and 15, which were also recognised by the majority of human experts but the standard explorer either did not recognise at all (e.g. command 8) or recognised only partly (e.g. command 15). For example, at command 8 of the sample session of Table 2, the user instead of pasting a file that s/he had cut previously, s/he issued a copy command. This resulted in a tricky situation where the user did not realise that s/he had made a mistake and s/he was running the risk of deleting the file which s/he erroneously believed that was moved elsewhere. This

mistake was also recognised by all of the human experts but was not recognised by the standard explorer at all, which considered the command correct. In this case, IFM's reasoning was compatible with a unanimous opinion of the human experts.

USER COMMANDS	**IFM'S REASONING**	**RESPONSES OF A STANDARD EXPLORER**	**COMPATIBILITY OF IFM'S ADVICE WITH THE HUMAN EXPERTS' ADVICE**
1. create_new_folder _in(A:\)	Neutral command.	No response.	
2. rename(A:\NewFolder, A:\java\)	Expected command concerning the renaming of the new folder.	No response.	
3. create_new_folder _in (A:\java1\)	Suspect command. IFM's alternative command: create_new_folder_in(A:\java\). This command is suggested because the newly created A:\java\ has no contents, whereas A:\java1\ already has a lot of contents.	No response.	IFM's reasoning was compatible with the opinion expressed by 30% of the human experts.
4. delete (A:\java1\New Folder\)	Neutral command.	Are you sure you want to remove the folder 'New Folder' and all its contents?	
5. create_new_folder _in (A:\java\)	Expected command. This command verifies IFM's suggestion in action 3.	No response.	
6. rename (A:\java\New Folder\, A:\java\programs\)	Expected command.	No response.	
7. cut(A:\java1\exerciseA\optional\exercise.java)	Neutral command.	No response.	

Table 2: A part of a user's protocol with IFM's comments and comparison with a standard explorer and human experts' comments

USER COMMANDS	IFM'S REASONING	RESPONSES OF A STANDARD EXPLORER	COMPATIBILITY OF IFM'S ADVICE WITH THE HUMAN EXPERTS' ADVICE
8. copy (A:\java\programs\)	Suspect command. IFM would expect a "paste" action following the "cut" action at command 5. IFM's alternative command: paste(A:\java\programs\). This command is suggested because copy is similar to paste and A:\java\programs\ has been selected by the user . Moreover, A:\java\programs\ is a newly created folder (on top of a stack of recently created folders) that has not been assigned any content yet.	No response.	IFM's reasoning about the possible error of the user and the need of a "paste" action was compatible with the opinion expressed by the majority of the human experts (100%).
9. create_new_folder_in (A:\)	Neutral command.	No response.	
10. rename(A:\New Folder\, A:\ java2\)	Neutral command.	No response.	
11. cut(A:\java1\ exerciseB\ user_manual.doc)	Neutral command.	No response.	
12. paste(A:\java2\)	Expected command.	No response.	
13. cut (A:\java1\exercise B\essay.doc)	Neutral command.	No response.	
14. paste(A:\java2\)	Expected command.	No response.	
15. deldir(A:\java1\ exerciseA\)	Suspect command. IFM's alternative command: deldir(A:\java1\exerciseB\) . This command is suggested because A:\java1\exerciseB\ has been left empty unlike A:\java1\exerciseA\. Moreover, the names of the folders are very similar and one could have been mistaken for the other.	Are you sure you want to remove the folder 'exerciseA' and all its contents?	IFM's reasoning was compatible with the opinion expressed by 70% of the human experts.

Table 2 cont.: A part of a user's protocol with IFM's comments and comparison with a standard explorer and human experts' comments

However, there were cases where there was a diversity of human experts' opinions. In those cases, IFM's advice was usually identical to the advice of the majority of human experts (e.g. in command 15 of the sample session of Table 2). In

this particular command IFM recognised an error, which was also recognised by 70% of the human experts. In this case, the user tried to delete a folder which had a lot of contents although in the previous commands 11-14 s/he had emptied a similar but different folder by moving all of its contents elsewhere. IFM in common with 70% of the human experts thought that the user probably really meant to delete the emptied folder rather than the one selected. However, the standard explorer only prompted the user with a generic confirmation message of the type: "Are you sure you want to remove the folder 'exerciseA' and all its contents?" Unlike IFM, it did not suggest any alternative command that the user may have meant instead of the one issued.

In fewer cases IFM's advice was compatible to the advice provided by a minority of experts (e.g. in command 3 of the sample session demonstrated in Table 2). However, it is worth noting that in the particular command the user's subsequent actions (4 and 5) verified the correctness of IFM's hypothesis.

In total, in the sample session IFM managed to follow the user's correct or incorrect reasoning successfully in 5 "expected" commands and 3 "suspect" commands which account for more than half of the commands of the session. In contrast, a standard explorer has no reasoning concerning commands, which are compatible or incompatible with the users' intentions and thus cannot follow the user's reasoning in the way that IFM can.

5.2 Summative Results

The users' protocols that were examined in this experimental study consisted of 1260 users' actions. In these actions, there were 135 erroneous or possibly erroneous actions according to a standard file manipulation program. All of these possibly erroneous actions, which accounted for 11% of the total actions, were prompted with the standard explorer's error and/or confirmation messages. In the 1260 actions, the majority of human experts identified 185 possibly unintended actions, which accounted for 15% of the total actions. These possibly unintended actions included the ones that were prompted with error messages by a standard explorer but they did not necessarily include the commands that were prompted with warning/or confirmation messages (e.g. command 4 in the sample session). In addition, they also included commands that were "correct" according to a standard explorer's formalities but were incompatible with the users' intentions (e.g. commands 8 and 15 of the sample session in Table 2). IFM recognised 226 possibly unintended actions, which accounted for 18% of the total actions.

Table 3 summarises the total actions that were considered as possibly unintended according to a standard explorer, IFM, a unanimous opinion of human experts, the majority of human experts and at least a minority of human experts respectively. Examples of such considerations can be found in the sample session of Table 2: Commands 4 and 15 were considered possibly unintended by a standard explorer (13% of the total actions of the sample session). Commands 3, 8 and 15 were considered possibly unintended by IFM (20% of the total actions of the sample session). Command 8 was considered possibly unintended by all of the human experts (6% of the total actions of the sample session). Commands 8 and 15 were considered possibly unintended by the majority of human experts (13% of the total actions of the sample session). Commands 3, 8 and 15 were considered possibly unintended by at least a minority of human experts (20% of the total actions of the sample session).

However, it must be noted that the actual percentages in Table 3 do not indicate the success or not of IFM and the standard explorer in comparison with human experts because these percentages do not show how compatible the views of IFM or the standard explorer were with the views of human experts. For example, in the sample session the standard explorer produced a warning message in two commands (command 4 and 15). This number is the same as the number of commands that were considered as unintended by the majority of human experts (command 8 and 15). However, the actual commands that alerted the majority of human experts were not identical to the commands that alerted the standard explorer. On the other hand, the commands that alerted the majority of human experts also alerted IFM.

In view of these quality differences, the degree of success of IFM and the standard explorer was measured by the degree of compatibility of their alert with that of the human experts. For example, in the sample session, the possibly unintended commands that were identified by both the majority of human experts and IFM were 2 (commands 8 and 15). In this session the compatibility of alert of IFM with the majority of human experts was 100%. On the other hand, in the sample session there was just one possibly unintended command (15) that was identified by both the majority of the human experts and the standard explorer although the majority of human experts had identified two commands (8 and 15). Hence in this session the compatibility of alert of the standard explorer with the majority of human experts was just 50%. Table 4 illustrates the rates of compatibility of alert of IFM and the standard explorer with human experts in the total 1260 actions of the experiment.

	All sessions	Sample Session of Table 2
Total Actions	1260	
Total possibly unintended actions according to the standard explorer	135 (11%)	13%
Total possibly unintended actions according to IFM	226 (18%)	20%
Total possibly unintended actions according to a unanimous opinion of human experts	96 (8%)	6%
Total possibly unintended actions according to the majority of human experts	185 (15%)	13%
Total possibly unintended actions according to at least a minority of human experts	252 (20%)	20%

Table 3: Summative results about considerations of users' possibly unintended actions

	IFM		Standard explorer	
Compatibility of alert in cases where the human experts had a unanimous opinion	Possibly unintended actions identified by IFM and all human experts	Possibly unintended actions identified by IFM and all human experts/Possibly unintended actions identified by all humans	Possibly unintended actions identified by a standard explorer and all human experts	Possibly unintended actions identified by standard explorer and all human experts/Possibly unintended actions identified by all humans
	88	88/96=92%	52	52/96=54%
Compatibility of alert with the majority of human experts	Possibly unintended actions identified by IFM and the majority of humans	Possibly unintended actions identified by IFM and the majority of humans/Possibly unintended actions identified by the majority of humans	Possibly unintended actions identified by standard explorer and the majority of humans	Possibly unintended actions identified by standard explorer and the majority of humans/Possibly unintended actions identified by the majority of humans
	164	164/185=89%	87	87/185=47%
Compatibility of alert with at least a minority of human experts	Possibly unintended actions identified by IFM and at least a minority of human experts	Possibly unintended actions identified by IFM and at least a minority of human experts/Possibly unintended actions identified by at least a minority of human experts	Possibly unintended actions identified by standard explorer and at least a minority of human experts	Possibly unintended actions identified by standard explorer and at least a minority of human experts /Possibly unintended actions identified by at least a minority of human experts
	218	218/252=87%	97	97/252=38%

Table 4: Comparison of the standard explorer and IFM with human experts' alert.

Concerning the comparison of IFM's responses to human experts' responses the results were very encouraging. The compatibility of the recognition of possibly unintended actions of IFM with the majority of human experts was 89%, whereas the respective degree for the standard explorer was just 47%. In cases where all human experts thought that the user's action was unintended, IFM had been alerted as well in 92% of these cases. However, the standard explorer was only alerted in 54% of these actions. Hence, IFM proved to be very successful in cases were there was a total agreement of human experts' opinions.

Compatibility of advice in cases where there was a unanimous opinion of human experts	Compatible advice of IFM with a unanimous opinion of human experts	Compatible advice of IFM with a unanimous opinion of human experts/ Possibly unintended actions identified by all humans	Compatible advice of standard explorer with unanimous of human experts	Compatible advice of standard explorer with unanimous of human experts /Possibly unintended actions identified by all humans
	88 commands	88/96=92%	17 commands	17/96=17.7%
Compatibility of advice with the majority of human experts	Compatible advice of IFM with the majority of humans	Compatible advice of IFM with the majority of humans/Possibly unintended actions identified by the majority of humans	Compatible advice of standard explorer with the majority of humans	Compatible advice of standard explorer with the majority of humans/Possibly unintended actions identified by the majority of humans
	153 commands	153/185=82.7%	17 commands	17/185=9%
Compatibility of advice with at least one human expert	Compatible advice of IFM with at least on human expert	Compatible advice of IFM with at least on human expert / Possibly unintended actions identified by at least a minority of human experts	Compatible advice of standard explorer with at least a minority of human experts	Compatible advice of standard explorer with at least on human expert/Possibly unintended actions identified by at least a minority of human experts
	188	188/252=75%	17	17/252=7%

Table 5: Comparison of the standard explorer and IFM with human experts' advice

However, even in cases where the same commands alerted both the human experts and the standard explorer, the content of the advice of these two parties was

not necessarily the same. For example, at command 15 of the sample session both the human experts and the standard explorer were alerted. Despite this fact, the majority of human experts would suggest the user to delete a particular folder other than the one mentioned in the command issued whereas the standard explorer would only produce a generic confirmation message. On the other hand, IFM produced the same content of advice as the majority of human experts in this particular command.

In view of these differences/similarities in the content of advice generated in the cases of alert we have compared IFM and the standard explorer in terms of the compatibility of their advice with that of the human experts (Table 5). In this aspect IFM proved to be much more similar to the human experts than the standard explorer.

The experiment revealed that IFM reacted in many more cases where the human experts had reacted as well than the standard explorer and produced much more similar advice to that of human experts. Furthermore, the value of the degree of compatibility of IFM's advice with human experts revealed that the system could successfully reproduce human experts' advice to a satisfactory extent. In particular, IFM was especially successful at recognising errors that looked quite obvious to a human advisor but were not recognised by a standard explorer at all. Such errors were recognised by all of the human experts. Recognising "obvious" human errors is a great improvement in a user interface in terms of its user-friendliness; this cannot be achieved if human reasoning is not incorporated into a system. However, even human advisors, in their minds, may only model an approximation of users' beliefs. Therefore, it was beyond the scope of IFM (and consequently of the evaluation) to produce reasoning that would exceed the capabilities of human experts who observed the interaction.

6 Evaluating User-IFM Interaction

Another usability test of IFM involved observing how novice and expert users interacted with IFM and how useful IFM could be in terms of recognising, diagnosing and preventing errors made by both expert and novice users.

6.1 The Experiment

For the usability evaluation of IFM, 16 users were selected. They had diverse backgrounds and interests and constituted a representative sample of expert and novice users. All 16 users were asked to interact with IFM, as they would normally do with a standard file manipulation program. Thus, in case IFM diagnosed a problematic situation, it informed the user that perhaps there was something wrong and suggested an alternative command.

The experiment required making observations about the users as they interacted with the system. Therefore, computer logging was used in order to register all users' action. The protocols collected were studied very carefully after the completion of the users' interaction with IFM and then the users were also interviewed so that they could give their own views bout what had happened during their interaction with the system.

A sample session of a user's interaction with IFM together with IFM's reasoning is illustrated in Table 6. The first column presents the user's commands and the second column illustrates IFM's reasoning. In case a command was considered

neutral or expected, it was executed normally. Otherwise, IFM informed the user that s/he was probably mistaken. The full reasoning of IFM together with the alternative commands that it suggested are presented in the second column of Table 6. Figure 12 illustrates the file store state of the particular user at the beginning of the sample session. The folder A:\18-7-2001\ contains three files, named "submitted.doc", "program.pas", "report.doc".

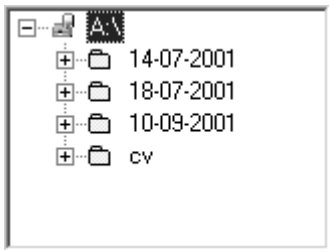

Figure 12: The user's initial file store state

In the example presented in Table 6, IFM succeeded at providing the right advice at command 7 where the user had accidentally selected "delete" instead of "rename" from the drop-down menu. Thus, IFM succeeded at preventing the user from issuing a catastrophic accidental deletion of a newly created folder where s/he had already moved 2 files and copied one file. Moreover, at command 12, IFM was rightfully alerted when the user attempted to paste the whole folder A:\18-7-2001\, into another folder, A:\15-10-2001\, where s/he had already pasted the contents of A:\18-7-2001\. IFM generated a plausible alternative by transforming the copy command and suggested that perhaps the user wanted to copy a folder other than the one selected with a very similar name.

However, the user had made a mistake in the paste command and had selected the wrong folder. Internally, IFM had generated this command as well but it selected to show to the user the one with the highest similarity to the command issued.

6.2 Analysis of the protocols

After the users had completed their interaction with IFM, the protocols were collected and were given to 5 human experts to comment on them. The human experts were asked to evaluate IFM's reasoning in the light of the users' actions. Therefore, IFM was evaluated in terms of how successful it was at following the users' reasoning and producing helpful advice. The success of IFM at following the users' reasoning was estimated by taking into account how often users' actions were considered as expected and how often users' actions were rightfully considered as suspect.

USER'S COMMANDS		IFM'S COMMENTS
1.	create_new_folder_in(A:\)	**Neutral command.**
2.	explore(A:\18-7-2001\)	**Neutral command.**
3.	cut(A:\18-7-2001\submitted.doc)	**Neutral command.**
4.	paste(A:\New Folder\)	**Expected command.** This command deletes the instability added in the previous action by the cut command.
5.	cut(A:\18-7-2001\program.pas)	**Neutral command.**
6.	paste(A:\New Folder\)	**Expected command.** This command deletes the instability added in the previous action by the cut command.
7.	delete(A:\New Folder\)	**Suspect command.** This command will result in losing many valuable files and folders. [IFM's suggestion: Did you mean to rename New Folder? Delete and rename are in neighbouring positions in the menu of commands.]
8.	rename(A:\New Folder\, A:\15-10-2001\)	**Expected command.** This command deletes the instability for the name New Folder of the newly created folder. This action verifies the correctness of IFM's suggestion in command 7.
9.	copy(A:\18-7-2001\report.doc)	**Neutral command.**
10.	paste(A:\15-10-2001\)	**Expected command.** This command deletes the instability added in the previous action by the copy command.
11.	copy(A:\18-7-2001\)	**Neutral command.**
12.	paste(A:\15-10-2001\)	**Suspect command.** A:\15-10-2001\ already contains a copy of the contents of A:\18-7-2001\. [IFM's suggestion: In the current command and the previous one, did you really mean what you have issued i.e. to copy A:\18-7-2001\ into A:\15-10-2001? Yes☐ No☐ Or perhaps you really meant to do one of the following: 1. Copy A:\14-7-2001 and paste it into A:\15-10-2001? Yes☐ No☐ 2. Other commands Yes☐ No☐ The user selects "Other commands" and issues the replacement command 13.
13.	paste(A:\10-9-2001\)	**Expected command.** This command deletes the instability added in action 11 by the copy command.
14.	delete(A:\18-7-2001\)	**Expected command.** This folder contains copies of files that can be found elsewhere and therefore its existence is pointless.

Table 6: A part of a user protocol with IFM's reasoning

In case an action was considered by IFM as expected it meant that IFM had been successful at recognising the user's intentions and following the user's reasoning. On the other hand, IFM was considered to have been rightfully alerted if it prompted the user with an "alert message" about an action being suspect and the user acknowledged his/her error by issuing a corrective action. Moreover, in cases when the corrective action issued by the user was the same as the one generated by IFM, then IFM was considered totally successful at producing helpful advice.

Total protocol actions	619	
	Actions	Percentage
Expected	195	195/619=32%
Suspect	88	88/619=14%
Neutral	294	294/619=47%
Erroneous	42	42/619=7%
Suspect + Erroneous	130	130/619=21%

Table 7: Categorisation of commands according to IFM

In the example of Table 6 there were 6 expected actions out of 14, which accounted for 42% of the total actions. Moreover, IFM was rightfully alerted in two actions (action 7 and action 12). If these 2 actions are added to the 6 expected actions then this accounts for 57% of the total actions where IFM was successful at following the user's reasoning and recognising his/her intentions. IFM was totally successful at producing helpful advice in action 7.

In all the collected protocols, which consisted of 619 actions, IFM categorised these actions in the way that is illustrated in Table 7. IFM was alerted in commands that it considered as suspect or erroneous. In total there were 130 actions that were considered suspect or erroneous which accounted for 21% of the total actions. In these actions, IFM produced messages containing advice to the user. IFM managed to recognise 195 (32%) actions as expected and was rightfully alerted in 102 (16%) actions. If we sum up these actions and divide them by the total actions of the users, we find out that IFM managed to follow 48% of the total actions of the users.

IFM was considered to have failed completely to produce helpful advice when it was not alerted at all in cases where a user had a problem. Indeed, IFM could not recognise all users' errors. There were cases where the users had made a mistake but IFM did not recognise it at all. This might have happened because IFM had not realised that the user's action was suspect. However, in most problematic situations, IFM did categorise the user's action as suspect but in some cases it could not find any alternative actions to propose to the user. In those cases the action was normally executed and the user suffered the consequences of his/her action. One way or another, IFM did not intervene in 22% of the users' actions that resulted in a state that was undesired by these users as it was revealed by the users' interviews.

Finally, IFM was considered to have been misleading when it was unnecessarily alerted. In general, IFM was unnecessarily alerted in approximately 5% of the total actions. Finally, there was a small percentage (10%) of the cases that IFM intervened, where the user admitted that s/he had made an error but his/her next action was not at all compatible with IFM's advice. However, this does not necessarily mean that IFM

had given the wrong advice; perhaps the users changed their plans after they were notified about their possible mistakes.

The above rates have been estimated as the average between the rates of novice and expert users. In fact, the rates of novice and expert users separately show that IFM has been more successful in the category of novice users than in the category of experts. Finally, we have observed that users learn from their own mistakes if there is a system advising them and the rate of error occurrence gradually decreases. This does not happen in standard file manipulation programs, as we have observed during the empirical evaluation. Moreover, through IFM's usage we have an increase of the cases where users reach their final goals and do not lose useful information that they cannot recover.

7 Discussion and Conclusions

In this paper, we have presented the software engineering approach that we have used for the development of an intelligent assistant of users' errors. The assistant developed, relies heavily on a novel adaptation of the cognitive theory Human Plausible Reasoning into a GUI that deals with file manipulation.

The multiple iterations of the software life cycle, were particularly useful for constructing and refining the knowledge-base of the system. To a large extent, this was achieved due to the multiple evaluations of the design that allowed the involvement of users and human advisors in experimental studies. Though multiple evaluations are also advocated by user interface designers [Sommerville, 92], [Dix et al., 93], [Shneiderman, 98], [Olsen, 98], in the case of knowledge-based user interfaces that incorporate user modelling techniques, evaluations are often neglected completely. This may have disasterous consequences on the system's overall credibility and effectiveness. On the other hand, multiple experimental studies allow the refinement and correction of the design at early stages of the development. Moreover, they render the knowledge elicitation and acquisition procedures more effective.

This was certainly the case in IFM where human advisors were asked to comment on user protocols and thus reveal many aspects of their reasoning at different stages of the development of the system. In this way, we specified what is missing from the reasoning of standard applications in comparison with human observers. Moreover, we specified fully the aspects of HPR, which were not completely specified originally and achieved an adaptation of the theory that was compatible with the human experts' reasoning and the users' reasoning.

Another advantage of the experiments that were conducted at several stages of the development, was the fact that the same experimental settings could be used in all phases. In particular, the methods that we used for evaluation were largely based on comparisons of the file manipulation programs with human experts' reasoning who were acting as observers of the interaction. In this way, we could approach the goal of rendering the interaction more human-like by eliciting and refining the knowledge and reasoning of a human observer of a human user of a GUI rather than the human user directly.

Overall, the results of the evaluation revealed that indeed users often encounter problems with the use of a standard explorer; therefore the existence of an intelligent

assistant would be needed. In addition, IFM's final release was quite successful at producing reasoning similar to the majority of human experts that took part in the evaluation. However, generally user modelling components of user interfaces may only model an approximation of users' beliefs. Therefore, it was beyond the scope of IFM to produce reasoning that would exceed the capabilities of human experts who observed the interaction.

Finally, the report on the development cycle of IFM clarifies issues related to the construction of intelligent assistance, such as what is aimed, what is achieved, how theories and models may be employed for the construction of software that will respond to real users' needs. Indeed, Delisle and Moulin [Delisle and Moulin, 02] after having reviewed the literature in user interfaces and help systems conclude that work on help systems would greatly benefit from in-depth studies of today's users' frustrations and expectations. The scarcity of similar reports in the literature of intelligent assistance renders this paper a source of know-how for the future developments in the field.

References

[Chin, 89] Chin D.N.: "KNOME: Modeling What the User Knows in UC", in Kobsa A. and Wahlster W. (eds.) User Models in Dialog Systems, 1989, 74-107.

[Chin, 01] Chin, D.N.: "Empirical Evaluation of User Models and User-Adapted Systems", User Modeling and User Adapted Interaction, 2001, Vol. 11, No. 1/2, 181-194.

[Collins and Michalski, 89] Collins, A., Michalski, R.: "The Logic of Plausible Reasoning: A core Theory", Cognitive Science, 1989, Vol. 13, 1-49.

[Delisle and Moulin, 02] Delisle, S., Moulin, B.: "User Interfaces and Help Systems: From Helplessness to Intelligent Assistance", in: P. Mc Kevitt (Ed.) Artificial Intelligence Review, 2002, Vol. 18, No. 2, 117-157.

[Dix et al., 93] Dix, A., Finlay, J., Abowd, G., and Beale, R.: "Human-Computer Interaction" NY: Prentice-Hall, 1993.

[Eller and Carberry, 92] Eller, R., Carberry, S.: "A Meta-rule Approach to Flexible Plan Recognition in Dialogue", in User Modeling and User-Adapted Interaction, 1992, Vol. 2, No. 1-2, 27-53.

[Eriksson and Penker, 98] Eriksson, H.E., Penker, M.: "UML Toolkit", John Wiley & Sons Inc, 1998.

[Höök, 00] Höök, K.: "Steps to take before intelligent user interfaces become real," Interacting with Computers, 2000, Vol. 12, 409-426.

[Horvitz et al., 98] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K.: "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users", Proceedings of the fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann: San Francisco, 1998, 256-265.

[Jerrams-Smith, 00] Jerrams-Smith J.: "An Intelligent Human-Computer Interface for Provision of On-Line Help", in: St.J. Hegner, P. Mc Kevitt, P. Norvig and R. Wilensky (Eds.) Artificial Intelligence Review, Intelligent Help Systems For Unix, 2000, Vol. 14, No. 1/2, 5-22.

[Kruchten, 99] Kruchten, P.: "Rational Unified Process-An Introduction", Addison-Wesley, 1999.

[Lif, 98] Lif, M.: "User Interface Modelling – adding usability to use cases", Rep. No 84 CMD, Uppsala University, 1998.

[Matthews et al., 00] Matthews, M., Pharr, W., Biswas, G., and Neelakandan: "USCSH: An Active Intelligent Assistance System," In Hegner, St.J., Mc.Kevitt, P., Norvig, P., and Wilensky, R. (Eds.) Artificial Intelligence Review, Intelligent Help Systems For Unix, 2000, Vol. 14, 121-141.

[Mayfied, 92] Mayfield, J.: "Controlling inference in plan recognition", in User Modeling and User-Adapted Interaction, 1992, Vol. 2, No. 1-2, 55–82.

[McTear, 00] McTear M.F.: "Intelligent interface technology: from theory to reality?", in Interacting with computers, 2000, Vol. 12, 323-336.

[Microsoft Corporation, 98] Microsoft Corporation: "Microsoft® Windows® 98 Resource Kit", Microsoft Press, 1998.

[Muller, Haslwanter and Dayton, 97] Muller, M.J., Haslwanter J.H., and Dayton, T.: "Participatory Practises in the Software Lifecycle", in Helander, M., Landauer, T.K. and Prabhu, P. (Eds.), Handbook of Human-Computer Interaction, Elsevier Science B.V., Amsterdam, 1997, 255-297.

[Muller, 97] Muller, P.A.: "Instant UML", Wrox Press Ltd, 1997.

[Olsen, 98] Olsen, Jr., D.R.: "Developing User Interfaces", Morgan Kaufmann Publishers, Inc, 1998.

[Quatrani, 98] Quatrani, T.: "Visual Modeling with Rational Rose and UML", Addison-Wesley, 1998.

[Quilici, 89] Quilici, A.: "AQUA: A system that detects and responds to user misconceptions", in A. Kobsa and A. Wahlster (eds.) User Modeling and Dialog Systems, New York, Springer-Verlag, 1989, 108-132.

[Rich, 99] Rich, E.: "Users are individuals: individualizing user models", International Journal of Human-Computer Studies, 1999, Vol. 51, 323-338.

[Shneiderman, 98] Shneiderman, B.: "Designing the User Interface: Strategies for Effective Human-Computer Interaction", Addison-Wesley, 1998.

[Sison & Simura 98] Sison R. and Shimura M.: "Student modeling and machine learning", International Journal of Artificial Intelligence in Education, 1998, Vol. 9, 128-158.

[Sommerville, 92] Sommerville, I.: "Software Engineering", Addison-Wesley, 1992.

[Sutcliffe, Ennis and Hu, 00] Sutcliffe, A.G., Ennis, M., and Hu J.: "Evaluating the effectiveness of visual user interfaces for information retrieval", International Journal of Human-Computer Studies, 2000, Vol. 53, No. 5, 741-763.

[Virvou, 99] Virvou, M.: "Automatic reasoning and help about human errors in using an operating system", Interacting with Computers, 1999, Vol. 11, No. 5, 545-573.

[Virvou and du Boulay, 99] Virvou, M., Du Boulay, B.: "Human Plausible Reasoning for Intelligent Help", User Modeling and User-Adapted Interaction, 1999, Vol. 9, 321-375

[Virvou, Jones and Millington, 00] Virvou, M., Jones, J., and Millington, M.: "Virtues and Problems of an Active Help System for UNIX", in: St.J. Hegner, P. Mc Kevitt, P. Norvig and R. Wilensky (Eds.) Artificial Intelligence Review, Intelligent Help Systems For Unix, 2000, Vol. 14, No. 1/2, 23-42.

[Virvou and Kabassi, 02] Virvou, M., Kabassi, K.: "F-SMILE: An Intelligent Multi-Agent Learning Environment," Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 2002), IEEE Computer Society, 2002, 144-149.

[Wilensky et al., 00] Wilensky, R., Chin, D.N., Luria, M., Martin, J., Mayfield, J., and Wu, D.: "The Berkeley UNIX Consultant Project," in: St.J. Hegner, P.Mc. Kevitt, P. Norvig and R. Wilensky (Eds.) Artificial Intelligence Review, Intelligent Help Systems For Unix, 2000, vol. 14, No 1-2, pp. 43-88.