Automated Generation of Agent Behaviour from Formal Models of Interaction

Fiorella de Rosis, Berardina De Carolis and Sebastiano Pizzutilo
Department of Informatics, University of Bari, Italy
{derosis,decarolis,pizzutilo}@di.uniba.it

ABSTRACT

We illustrate how a formal model of interaction can be employed to generate documentation on how to use an application, in the form of an Animated Agent. The formal model is XDM, an extension of Coloured Petri Nets that enables representing user-adapted interfaces, simulating their behaviour and making pre-empirical usability evaluations. XDM-Agent is a personality-rich animated character that uses this formal model to illustrate the role of interface objects and to explain how tasks may be performed; its behaviour is programmed by a schema-based planning followed by a surface generation, in which verbal and non-verbal acts are combined appropriately; the agent's 'personality' may be adapted to the user characteristics.

Keywords

Animated agents, formal models of interaction, user manuals generation.

1. INTRODUCTION

The increasing complexity of user interfaces requires specific methods and tools to design and describe them; an emerging solution is to employ formal methods for a precise and unambiguous specification of interaction. Graphical methods are preferred since they are more easily perceived by users without a particular experience [6, 14, 28]. However, formal methods require a considerable effort in building the model; this is, probably, the main reason why even those of them that proved to be valid in HCI research find difficulties in being applied in interface engineering.

There are, currently, two directions in which research tries to overcome this problem issue. The first one aims at developing tools that simplify the modelling process by integrating formal methods with artificial intelligence techniques: the model is augmented with a knowledge base that represents HCI design guidelines, to generate semiautomatically the interface prototype: see, for instance, the MECANO [22] and MOBI-D [29] projects. The second direction of research is focused on proving that efforts spent in building a formal model can be partially got back if that model is employed to ease some of the designer's tasks, such as early prototyping, automated generation of interface objects and help messages and pre-empirical interface. Examples of projects that follow this trend are TADEUS [13] and TLIM [29]. A third perspective sees using knowledge in the formal model for generating software documentation. Documentation may refer to several aspects of software and may be addressed to several types of users. It may be aimed at reconstructing "logic, structure and goals that were used in writing a program in order to understand what the program does and how it does it", as in MediaDoc [12]; in this case, software engineers are the main users of the documentation produced. Alternatively, it may be aimed at describing how a given application can be used. In this case, the addressees of documentation are the end users of the application, whose need for information varies according to the tasks they perform and to their frequency of use and experience with of the application.

The idea of producing a User Manual as a byproduct of interface design and implementation becomes more practicable if a formal model of interaction is employed as a knowledge base to the two purposes. The most popular formal models and tools that had originally been proposed to guide interface design and implementation have been employed to automatically produce help messages: for instance, Petri Nets [25] and HUMANOID Hyper Help [22]; in these systems, help messages are presented as texts or hypertexts, in a separate window. To complete the software documentation, animations have been proposed as well (for example, in UIDE), that combine audio, video and demonstrations to help the user to learn how to perform a task [32]. Other projects focused on the idea of generating, from a knowledge base, the main components of an instruction manual [34, 24]: for instance, DRAFTER and ISOLDE's aim is to generate multilingual manuals from a unique knowledge base [27, 31, 17]. Some of these Projects start from an analysis of the manuals of some well-known software products, to examine the types of information they include and the linguistic structure of each of them [15]. By adopting the metaphor of 'emulating the ideal of having an expert on hand to answer questions', I-Doc (an Intelligent Documentation production system) analyses the interactions occurring during expert consultations, to categorize the users' requests and to identify the strategies they employ for finding the answer to their question issues [18]. This study confirms that the questions users request are a function not only of their tasks, but also of their levels of experience: more system-oriented questions are asked by novices while experts tend to ask goal-oriented, more complex questions.

With the recent spread of research on animated characters, the idea of emulating, in a User Manual, the interaction with an expert, has a natural concretisation in implementing such a manual in the form of an Animated Agent. The most notable examples of Pedagogical Agents are Steve, Adele, Herman the Bug, Cosmo and PPP-Persona, all aimed at some form of intelligent assistance, be it presentation, navigation on the Web, tutoring or alike [19, 2, 21, 30, 4]. Some of these Agents combine explanation capabilities with the ability to provide a demonstration of the product, on request.

In previous papers, we proposed a formalism named XDM (Context-Sensitive Dialogue Modeling), in which Coloured Petri Nets are extended to specify user-adapted interaction modeling; we then described a tool for building XDM models and simulating the interface behavior, and we enhanced this tool with ability to perform pre-empirical evaluations of the interface correctness and usability [8]. We subsequently investigated how these models could be used as a knowledge source for generating on-line user manuals in the form of hypermedia or of an 'Animated Pedagogical Agent' that we called 'XDM-Agent': in this paper, we present the first results of this ongoing Project. In the following sections, after justifying why we selected an Animated Agent as a presentation tool, we describe the XDM-Agent's main components; we then discuss limits and interests of this approach and conclude with some comparison with related works.

2. THE ANIMATED AGENT APPROACH

As a first step of our research on software documentation, we studied how XDM-Models could be employed to generate various types of hypertextual help, such as: 'which task may I perform?', 'how may I perform this task? 'why is that interaction object inactive?', and others; we employed schema and ATN-based natural language generation techniques to produce the answers. Shifting from hypertextual helps to an Agent-based manual entails several advantages and raises several methodological problems. The main opportunity offered by Animated Agents is to see the software documentation as the result of a 'conversation with some expert in the field'. In messages delivered by Animated Agents, verbal and non-verbal expressions are combined appropriately to communicate information: this enables the documentation designer to select the media that is most convenient to vehicle every piece of information. As several media (speech, body gesture, face expression and text) may be presented at the same time, information may be distributed among the media and some aspects of the message may be reinforced by employing different media to express the same thing, to make sure that the user remembers and understands it. A typical example is deixis: when helps are provided in textual form, indicating unambiguously the interface object to which a particular explanation refers is not easy; an animated agent that can overlap to the application window may solve this problem by moving towards the object, pointing at it, looking at it and referring to it by speech and/or text. A second, main advantage, is in the possibility of demonstrating the system behavior (after a 'How-to' question) by mimicking the actions the user should do to perform the task and by showing the effects these actions will produce on the interface: here, again, gestures may reinforce natural language expressions. A third advantage is in the possibility of making visible, in the Agent's attitude, the particular phase of dialog: by expressing 'give turn', 'take turn', 'listening', 'agreeing', 'doubt' and other meta-conversational goals, the Agent may give the users the impression that they are never left alone in their interaction with the documentation system, that this system really listens to them, that it shows whether their question was or wasn't clear,… and so on.

However, shifting from a hypermedia to an agent-based user manual corresponds to a change of the interaction metaphor that implies revising the generation method employed. In hypermedia, the main problems where to decide 'which information to introduce in every hypermedia node', 'which links to further explanations to introduce', 'which media combination to employ to vehicle a specific message', in every context and for every user. In agent-based presentations, the 'social relationship metaphor' employed requires reconsidering the same problems in different terms; it has, then, to be established 'which is the appropriate agent's behaviour' (again, in every context and for every user), 'how can the agent engage the users in a believable conversation' by providing, at every interaction turn, the 'really needed' level of help to each of them, 'how should interruptions be handled and user actions and behaviours be interpreted' so as to create the impression of interacting online with a tool that shares some of the characteristics of a human helper. These problems are common to all Animated Agents: in our project, we have examined how they may be solved in the particular case of software documentation.

3. XDM-AGENT

XDM-Agent is a personality-rich animated character that uses several knowledge sources to explain to the user how to use the application. The behaviour of this agent is programmed by a schema-based planning (the agent's 'Mind'), followed by a surface generation (its 'Body'), in which verbal and nonverbal acts are combined appropriately. The agent's personality, that is the way its Mind is programmed and its Body appears to the user, is adapted to the user characteristics. XDM-Agent exploits three knowledge sources: (i) a *formal description of the application interface*, (ii) a *description of the strategi*es that may be employed in generating the explanation and (iii) a *description of mental models of the two agents* participating in the interaction (the User and XDM-Agent). Let us describe in more detail these sources, and how they are used for generating the agent behaviour.

3.1 The Interface Description Formalism

XDM is a formalism that extends *Coloured Petri Nets* (CPN: [35]) to describe user-adapted interfaces. A XDM model includes the following components:

- a description by abstraction levels of how *tasks* may be decomposed into complex and/or elementary subtasks (with Petri Nets), with the relations among them;
- a description of the way elementary tasks may be performed, with a *logical and physical projection* of CPN's transitions; this description consists in a set of tables that specify the task associated with every transition, the action the user should make to perform it and the interaction object concerned;
- a description of the *display status* before and after every task is performed, with a 'logical and physical projection' of

CPN's places; this consists in a set of tables that specify information associated with every place and display layout in every phase of interaction.

To model user-adaptation, conditions are attached to transitions and to places, to describe when and how a task may be performed and how information displayed varies in every category of users. This allows the designer to restrict access to particular tasks to particular categories of users and to vary the way in which tasks may be performed and the display appears. A detailed description of this formalism may be found in [8], where we show how we used it in different projects to design and simulate a system interface and to make semiautomatic evaluations of consistency and complexity.

In the generation of the Animated User Manual, we employ a simplified version of XDM, in which Petri Nets (PNs) are replaced with UANs (User Action Notations: [25]). Like PNs, UANs describe tasks at different levels of abstraction: a UAN element represents a task; temporal relations among tasks are specified in terms of a few 'basic constructs':

sequence (A° B), iteration (A)*, choice (A | B), order independence (A & B), concurrency (A || B), interruptability (B $\rightarrow$ A) and interleavability (A $\Leftrightarrow$ B).

These constructs may be combined to describe the decomposition of a task T as a string in the alphabet that includes UAN elements and relation operators.

For example: T = (A ° (B | C))* ° D
indicates that subtasks B and C are in alternative, that A has to be performed before them, that the combination of tasks A, B and C may be iterated several times and that, finally, the task T must be concluded by the subtask D. Notice that subtasks A, B, C and D may be either elementary or complex; at the next abstraction level, every complex task will be described by a new UAN.

A UAN then provides a linearised, string-based description of the task relationships that are represented graphically in a Petri Net. Elements of UANs correspond to PNs' transitions; their logical and physical projections describe how tasks may be performed; conditions attached to UANs' elements enable defining access rights.

Figure 1 shows a representation of the knowledge base that describes a generic application's interface, as an Entity-Relationship (E-R) diagram. UANs' elements, Tasks (either complex or elementary), Interface Objects (again, complex or elementary) and Events are the main entities. Elementary interface objects in a window may be grouped into complex objects (a toolbar, a subwindow etc). A Task is associated with a (complex or elementary) Interface Object; a UAN describes how a complex task may be decomposed into subtasks and the relations among them; an elementary task may be performed by a specific Event on a specific elementary Interface Object; an elementary Interface Object may open a new window that enables performing a new complex task. Adaptivity is represented, in the E-R diagram, through a set of user-related conditions attached to the entities or to the relations (we omit these conditions from Figure 1 and from the example that follows, for simplicity reasons). A condition on a task defines access rights to that task; a condition on an object defines the user category to which that object is displayed; a condition on the task-object-event relation defines how that task may be performed, for that category of users, …and so on.

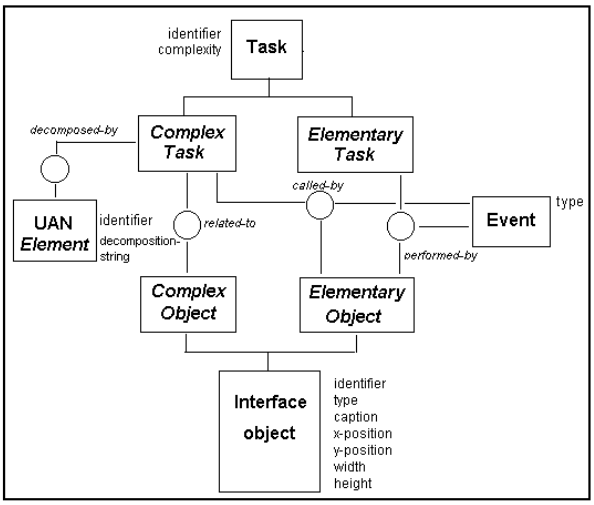


Figure 1. E-R representation of the application-KB.

Let us reconsider the previous example of UAN. Let Ti be a UAN element and UAN(Ti) the string that describes how Ti may be decomposed into subtasks, in the UAN language. Let Task(Ti), Obj(Ti) and Ev(Task(Ti), Obj(Ti)) be, respectively, the task associated with Ti, the interface object and the event that enable the user to perform this task. The application-KB will include, in this example, the following items:
UANs' elements: T, A, B, C, D
Complex Tasks:Task(T): database management functions
Task (A): input identification data
Elementary Tasks: Task (B): delete record, Task (C): update record, Task (D): exit from the task
UAN(T): (A ° (B | C))* ° D
Interface Objects: Obj (T): W1, Obj (A): W2, Obj (B): B1. Obj (C): B2, Obj (D): B3
Events: Ev(Task(D), Obj (D)): double-click
This model denotes that database management functions (that can be performed in window W1) need first to input identification data (with window W2), followed by deleting or updating a record (buttons B1 and B2); this combination of tasks may be repeated several times. One may, finally, exit from this task by double-clicking on button B3.

3.2 XDM-Agent's behaviour strategies

XDM-Agent illustrates the graphical interface of a given application starting from its main window or from the window that is displayed when the user requests the agent's help. The generation of an explanation is the result of a three-step process: a *planning* phase that establish the presentation content; a *plan revision* phase that produces a less redundant plan; a *realisation* phase, that translates the plan into a presentation. The hierarchical *planning* algorithm establishes how the agent will describe the main elements related to the window by reading its description in the application-KB: a given communicative goal, fired from the user request, is recursively decomposed into subgoals until 'primitive goals' are reached, that do not admit further decomposition. This process generates a tree structure whose leaves represent macro-behaviours that can be directly executed by the agent. At the planning level, adaptation is made by introducing personality-related conditions in the ***constraint***

field of plan operators, so that the same communicative goal may produce different decompositions in the different contexts in which the agent will operate. The presentation plan includes, most of the times, redundancies due to the fact that objects in the interface may be of the same type and tasks associated with them can be activated with the same interaction technique; an aggregation algorithm synthesizes common elements to produce a less repetitive presentation. Once the revised presentation plan for a window is ready, this is given as input to a *realisation* algorithm, that transforms it in a sequence of 'macro-behaviors' (a combination of verbal and nonverbal communicative acts that enables achieving a primitive goal in the plan).

The *list of macro-behaviors* that XDM-Agent agent is able to perform is application-independent but domain-dependent; it is the same for any application to be documented but is tailored to the documentation task:

a. *perform meta-conversational acts*: introduce itself, leave, take turn, give turn, make questions, wait for an answer;
b. *introduce-a-window* by explaining its role and its components;
c. *describe-an-object* by showing it and mentioning its type and caption (icon, toolbar or other);
d. *explain-a-task* by mentioning its name;
e. *enable-performing-an-elementary-task* by describing the associated event;
f. *demonstrate-a-task* by showing an example of how the task may be performed;
g. *describe-a-task-decomposition* by illustrating the relationships among its subtasks.

A macro-behaviour is obtained by combining verbal and non-verbal acts as follows:

- *verbal acts* are produced with natural language generation functions, that fill context-dependent templates with values from the application-KB; the produced texts are subsequently transformed into 'speech' or 'write a text in a balloon';
- *nonverbal acts* are 'micro behaviours' that are produced from MS-Agent's animations, with the aid of a set of auxiliary functions. The list of micro behaviours that are employed in our animated user manual is shown in **Figure 2.**

a. *Greetings*
Self_Introduction, Leave
b. *Meta-Conversational-Gestures*
Take_Turn, Give_Turn, Questioning, Listening
c. *Locomotive-Gestures*
Move_To_Object (Oi), Move_To_Location (x, y)
d. *Deictic-Gestures*
Point_At_Location (x,y), Point_At_Object (Oi)
Point-At_Area ((xi, yi), (xj, yj))
e. *Relation-Evoking*
Evoke_Sequence, Evoke_Iteration,
Evoke_Order_Independence, ……
f. *Event-Mimicking*
Mimic_Click,Mimic_Double_Click, Mimic_Keyboard_Entry,.
g. *Looking*
Look_At_User, Look_At_Location (x,y), Look_At_Object (Oi), Look_At_Area ((xi, yi), (xj, yj))
h. *Approaching-the-user*

Figure 2. Library of XDM-Agent's 'micro behaviours'

This list includes: (i) *object-referring gestures*: the agent may move towards an interface object or location, point at it and look at it; (ii) *iconic gestures*: the agent may evoke the relationship among subtasks, that is a sequence, a iteration, a choice, an order independence, a concurrency and so on; it can mimic, as well, the actions the user should do to perform some tasks: click, double click, keyboard entry,…and so on; (iv) *user-directed gestures*: the agent may look at the user, get closer to him or her by increasing its dimension, show a questioning or listening attitude, manifest its intention to give or take the turn, open and close the conversation with the user by introducing itself or saying goodbye. The way these micro behaviors are implemented depends on the animations included in the employed software: in particular, a limited overlapping of gestures can be made in MS-Agent[1], which only enables generating speech and text at the same time. We therefore overlap verbal acts to nonverbal ones so that the Agent can simultaneously move, speech and write something on a balloon, while we sequence nonverbal acts so as to produce 'natural' behaviors. We employ nonverbal acts to reinforce the message vehicled by verbal acts. So, the agent's speech corresponds to a self-standing explanation, that might be translated into a written manual; text balloons mention only the 'key' words in the speech, on which users should focus their attention; gestures support the communication tasks that could not be effectively achieved with speech (for instance, deixis), reinforce concepts that users should not forget (for instance, task relations), support the description of the way actions should be done (for instance, by mimicking events) and, finally, give the users a constant idea of 'where they are, in the interactive explanation process (for instance, by taking a 'listening' or 'questioning' expression). Finally, like for all embodied characters, speech and gestures are employed, in general, to make interaction with the agent more 'pleasant' and to give users the illusion of 'interacting with a companion' rather than 'manipulating a tool'.

3.3 The Mental Models

The two agents involved in interaction are the User and XDM-Agent. While the user modeling component is very simple (we classify the users according to their experience with the application), XDM-Agent's model is more interesting, since its behaviour is driven by some personality traits that we describe in terms of its 'helping attitude'. Let's see this in more detail.

A typical software manual includes three sections [15]: (i) a *tutorial*, with exercises for new users, (ii) a series of step by step *instructions for the major tasks* to be accomplished and (iii) a ready-reference *summary of commands*. To follow the 'minimal manual' principle ("the smaller the manual, the better": J M Carroll, cited [1]), the Agent should start from one of these components, provide the "really needed minimal" and give more details only on the user's request. The component from which to

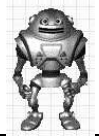
[1] MS-Agent is a downloadable software component that displays an animated character on top of an application window and enables it to talk and recognise the user speech. A character may be programmed by a language that includes a list of 'animations' (body and face gestures): these animations are the building blocks of XDM-Agent.

start and the information to provide initially may be fixed and general or may be varied according to the user and to the context. In the second case, as we mentioned in the Introduction, the user goals, his/her level of experience and his/her preference concerning the interaction style may drive selection of the Agent's "explanation attitude". Embodiment of the Agent may be a resource to make this attitude explicit to the user, by varying the Agent's appearance, gesture, sentence wording and so on.

XDM-Agent is able to apply two different approaches to interface description; in the *task-oriented* approach, it systematically instructs the user on the tasks the window enables performing, how they may be performed and in which sequence and provides, if required, a demonstration of how a complex task may be performed. In *command-oriented* descriptions, it lists the objects included in the window in the order in which they are arranged in the display and provides a minimal description of the task they allow to perform; other details are given only on the User request. Therefore, in the first approach the Agent takes the initiative and provides a detailed explanation, while in the second one the initiative is 'mixed' (partly of the Agent, partly of the User), explanations are, initially, less detailed and a dialog with the User is established, to decide how to go on in the explanation.

If the metaphor of 'social interaction' is applied to the User-Agent relationship, the two approaches to explanation can be seen as the manifestation of two different 'help personalities' in the Agent [9]: (i) a *overhelper*, that tends to interpret the implicit delegation received by the user in broad terms and explains anything he presumes the user desires to know, and (ii) a *literal helper*, that provides a minimal description of the concepts the user explicitly asks to know. These help attitudes may be seen as particular values of the dominant/submissive dimension of interpersonal behaviour[2], which is considered to be the most important factor affecting human-computer interaction [23].

Although some authors proved that dominance may be operationalised by only manipulating the phrasing of the texts shown in the interface and the interaction order (again, [23]; but also [10]), others claim that the user appreciation of the interface personality may be enhanced by varying, as well, the Agent's 'external appearance': body posture, arm, head and hands gestures, moving [16, 3, 5]. By drawing on the cited experiences, we decided to embody the overhelping, dominant attitude in a more 'extroverted' agent that employs a direct and confident phrasing and gestures and moves much. We embody, on the contrary, the literal helper, submissive attitude in a more 'introverted' agent, that employs lighter linguistic expressions and moves and gestures less. To enhance matching of the Agent's appearance with its underlying personality, we select *Genie* to represent the more extroverted, dominant personality and *Robby* to represent the submissive one]: this is due in part to the way the two characters are designed and animated, in the MS-Agent tool, and in part to the expectation they raise in the user: Genie is seen as more 'empathetic', someone who takes charge of the Users and anticipates their needs, while Robby is seen as more 'formal', someone who is there only to respond to orders. **Figure 3** summarises the main differences between the two characters.



Robby	**Genie**
Object-oriented presentation	*Task-oriented presentation*
submissive	Dominant
introverted	Extroverted
is rather 'passive'; says the minimum and waits for the user's orders	is very 'active':takes the initiative and provides detailed explanations
employs 'light' linguistic expressions, with indirect and uncertain phrasing (suggestions)	employs 'strong' linguistic expressions, with direct and confident phrasing (commands)
gestures the minimum: minimum locomotion, limited movements of arms and body, avoids getting close to the user	gestures are more 'expansive': more locomotion, wider movements, gets closer to the user
speaks slow	speaks high

Figure 3. Personality traits of Robby and Genie, with differences in behaviour.

How do we combine the Agent's personality with the User's characteristics? Some authors claim that task-based explanations would be more suited to novice users and object-oriented explanations more suited to experts. For instance, empirical analysis of a corpus of documents, in TAILOR, showed that complex devices are described in an object-oriented way in adult encyclopædias, while descriptions in junior encyclopædias tend to be organized in a process-oriented, functional way [26]. On the other side, a 'dominant', 'extroverted' personality is probably more suited to a novice, while a 'submissive' and 'formal' one will be more easily accepted by an expert user. This led us to select *Robby* for experts and *Genie* for novices.

4. IMPLEMENTATION

We implemented XDM-Agent in Java and Visual Basic, under Windows95. Adaptation of the generated documentation to the agent's personality is made at both the planning and the realisation phase. At the planning level, adaptation is made by introducing personality-related conditions in the planning schemas in order to generate a *task-oriented* or an *object-oriented* plan. At the surface realisation level, a unique Behaviour Library is employed in the two cases, with different ways of realising every behaviour. In the task-oriented plan, a window is introduced by mentioning the complex task that this window enables performing; the way this complex task is decomposed into less complex subtasks is then described, by examining (in the application-KB) the UAN associated with this window. For each element of the UAN, the task and the associated object are illustrated; if the task is 'elementary', the event enabling to perform it is mentioned; if it is is complex, the user is informed that a demonstration of how to perform it may be provided, if requested. Task relations (again, from the UAN) are then illustrated. Description goes on by selecting the next

[2] According to [23], dominance is marked, in general, by a behavior that is "self-confident, leading, self-assertive, strong and take-charge"; submissiveness is marked, on the contrary, by a behavior that is "self-doubting, weak, passive, following and obedient".

window to describe, as one of those that can be opened from the present window. In the object-oriented plan, interface objects are described by exploring their hierarchy in a top-down way; for each elementary object, the associated task is mentioned. The turn is then given to the user, who may indicate whether and how to proceed in the explanation.

Planning is totally separated from realisation: we employ, to denote this feature, the metaphor of 'separating XDM-Agent's *Mind* from its *Body'*; we might associate, in principle, Robby's appearance and behaviour to a task-oriented plan (that is, Robby's Body to Genie's Mind) and the inverse. This separation of the agent's body from its mind gives us the opportunity to implement the two components on the server-side and on the client-side respectively, and to select the agent's appearance that is preferable in any given circumstance. An example of the difference in the behaviour of the two agent's personalities is shown in **Figure 4, a** and **b.**

Macro-Behaviour	***Speech***	***Balloon***	***Gesture***
Introduce-a-window (task-oriented)	In this window, you can perform the main database management functions:		PointAtArea LookAtUser
Explain-task Describe-object	- to select a database management function, use the commands in this toolbar;	Select database managem. functions	MoveToObject PointAtObjec LookAtUser
idem	- to input identification data, use the textfields in this subwindow;	Input identificati on data	idem
	…….	…….	…….
Introduce-complex-task	There are two database management functions you may select:		
Explain-task Describe-object Enable-to-perform-a-task	- to update an existing record, click on the 'Update' button;	Update	MoveToObject PointAtObject LookAtObject LookAtUser MimicClick
idem	- to delete a record, click on the 'Delete' button;	Delete	idem

Figure 4 a: task-oriented description, by Genie

Macro-Behaviour	***Speech***	***Balloon***	***Gesture***
Introduce-a-window (object-oriented)	I'm ready to illustrate you the objects in this window:		LookAtUser
Introduce-complex-object	- a toolbar, with 5 buttons:		
Describe-object Explain-task	the first one enables you to update an existing record	Update	PointAtObject LookAtObject
idem	the second one enables you to deletea record	Delete	idem
	…..		

Figure 4b: object-oriented description, by Robby

5. INTERESTS AND LIMITS

The first positive result of this research is that we could verify that the formal model we employed to design and evaluate the interface (XDM) enables us, as well, to generate the basic components of an Animated Instruction Manual. Planning the structure of the presentation directly from this formal model contributes to insure that "the manual reflects accurately the system's program and that it can be viewed as a set of pre-planned instructions"[1]. An Animated User Manual such as the one we generate is probably suited to the needs of 'novice' users; we are not sure, on the contrary, that the more complex questions an expert makes can be handled efficiently with our application-KB. We plan to check these problems in an evaluation study, from which we expect some hints on how to refine our system. In this study, we plan to assess which is the best matching between the two XDM-Agent's personalities and the users characteristics, including their experience and their personality: in fact, the evidence on whether complementarity or similarity-attraction holds between the system and the user personalities is rather controversial [16], and we suspect that the decision depends on the particular personality traits considered.

The present prototype has several limits: some of them are due to the generation method we employ, others to the tool:

- The main limit in our generation systems is that we do not handle interruptions: user can make questions to the system only when the agent gives them the turn;
- the second limit originates from available animations. In MS-Agent characters, the repertoire of gestures is rather limited, especially considering face and gaze expressions. In addition, the difficulty of overlapping animations does not allow us to translate into face or body gestures the higher parts of the discourse plan; XDM-Agent thus lacks of those gestures that aid in integrating adjacent discourse spans into higher order groupings [20], for instance by expressing rhetorical relations among high-level portions of the plan. To overcome these limits, we should build our own character, with the mentioned animations.

We have, still, to assess whether Animated Agents really contribute to make software documentation more usable, in which conditions and for which user categories. This consideration applies to the majority of research projects on Animated Agents, which has been driven, so far, by an optimistic attitude rather than a careful assessment of the validity of results obtained.

5. RELATED WORK

Our research lies in the crossroad of several areas: formal models of HCI, user adaptation and believable agents. There are, we believe, some new ideas in the way these areas are integrated into XDM-Agent: we showed, in previous papers, that a unique formal model of HCI can be employed to unify several steps of the interface design and implementation process: after analysis of user requirements has been completed, these requirements can be transferred into a UI specification model that can be subsequently employed to implement the interface, to simulate its behavior in several contexts and to make pre-empirical usability evaluations. In this paper, we show that the same model

can be employed, as well, to produce an online user manual. Any change in the UI design must be transfered into a change in the formal model and automatically produces a new version of the interface, of the simulation of its behavior, of usability measures and of the documentation produced.

Adaptation to the user and to the context is represented through parameters in the model and reflects into a user-adapted documentation. In particular, adaptation to the user needs about documentation is performed through the metaphor of 'changing the personality of the character who guides the user in examining the application'. That computer interfaces have a personality was already proved by Nass and colleagues, in their famous studies in which they applied to computers theories and methods originally developed in the psychological literature for human beings (see, [23]). Taylor and colleagues' experiment demonstrates that a number of personality traits (in the 'Five-Factor' model) can be effectively portrayed using either voice alone or in combination with appropriately designed animated characters" [33]. Results of the studies by these groups oriented us in the definition of the verbal style and the nonverbal behaviors that characterise XDM-Agent's personalities (see, [16]); Walker and colleagues research on factors affecting the linguistic style guided us in the diversification of the speech attitudes [36]. We examined, in the past, the personality traits that might be relevant in HCI by formalising the cooperation levels and types and the way they may combine [7, 9]: Robby and Genie are programmed according to some of these traits; other traits (such as a critical helper, a supplier and so on) might be attributed to different characters, with different behaviors. The long-term goal of our research is to envisage a human-computer interface in which the users can settle, either implicitly or explicitly, the 'helping attitude' they need in every application: XDM-Agent is a first step towards this direction.

6. REFERENCES

[1] Addison, M. and Thimbleby, H. Social impacts of computing: Codes of professional ethics. *Social Science Computing Review*, 2 (Winter 1992), 453-469.

[2] André, E., Rist, T. and Mueller, J. Integrating reactive and scripted behaviours in a life-like presentation agent. In K P Sycara and M Woolridge (Eds*): Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, 1998.

[3] Arafa, J., Charlton, A., Mamdani and Fehin, P. Designing and building Personal Service Assistants with personality. *Proceedings of the Workshop on Embodied Conversational Characters*, S Prevost and E Churchill (eds), Tahoe City, 1999.

[4] Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., Stankosky, A., Thiel, D., van Dantzich, M. and Wax, T. Lifelike computer characters: the Persona project ar Microsoft. In J M Bradshaw (ed): *Software Agents.* AAAI/MIT Press, Menlo Park, 1997.

[5] Ball, G. and Breese, J. Emotion and personality in a conversational character. *Proceedings of the Workshop on Embodied Conversational Characters*, S Prevost and E Churchill (Eds), Tahoe City, 1999.

[6] Carneiro, L.M.F., Cowan, D.D and Lucena, C.J.P. (1994). ADV Charts: a visual formalism for interactive systems. *SIGCHI Bulletin*, 26, 2, 74-77.

[7] Castelfranchi, C., de Rosis, F., Falcone, R. and Pizzutilo, S. Personality traits and social attitudes in multiagent cooperation. *Applied Artificial Intelligence*, 12, 1998.

[8] de Rosis, F., Pizzutilo, S. and De Carolis, B. Formal description and evaluation of user-adapted interfaces. *International Journal of Human-Computer Studies*, 1998.

[9] de Rosis, F. and Castelfranchi, C. How can personality factors contribute to make agents more 'believable'? *Proceedings of the Workshop on 'Behaviour Planning for Life-like Characters and Avatars'*, Sitges, 1999.

[10] Dryer, D.,C. *Dominance and valence: a two-factor model for emotion in HCI*. AAAI Press, 1998.

[11] Elwert, T. and Schlungbaum, E. Modelling and Generation of Graphical User Interfaces in the TADEUS Approach, In: P. Palanque, R. Bastide (Eds.): *Designing, Specification, and Verification of Interactive Systems*. Wien, Springer, 1995, S. 193-208.

[12] Erdem, A., Johnson, W. L. and Marsella, S. Task-oriented software understanding. *Proceedings of the Automated Software Engineering Conference*, 1998.

[13] Eriksson, H., Puerta, A.R. and Musen, M. Generation of knowledge-acquisition tools from domain ontologies. *8th Knowledge-Acquisition for Knowledge-Based Systems Workshop,* 1994.

[14] Harel, D. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8, 231-274, 1987.

[15] Hartley, A. and Paris, C.Two sources of control over the generation of software instructions*. Proceedings of ACL 96*, 1996.

[16] Isbister, C. and Nass, C. Personality in conversational characters: building better digital interaction partners using knowledge about human personality preferences and perceptions. *Proceedings of the Workshop on Embodied Conversational Characters*, S Prevost and E Churchill (Eds), Tahoe City, 1999.

[17] Isolde: http://www.syd.dit.csiro.au/staff/cecile/isolde.html

[18] Johnson, W., L. and Erdem, A. Interactive explanations of software systems. *Automated Software Engineering, an International Journal.* 1996.

[19] Johnson, W.,L. Pedagogical Agents. http://www.isi.edu/isd/carte/ped_agents/pedagogical_agents.html

[20] Kendon, A. Some relationships between body motion and speech. In: *Studies in dyadic communication.* A W Siegman and B Pope (Eds), Pergamon, 1972.

[21] Lester, J.C., Voerman, J.L., Towns, G. and Callaway G. B. Deictic believability: coordinated gesture, locomotion and

speech in lifelike pedagogical agents. *Applied Artificial Intelligence*, 13 (4-5), 1999.

[22] Moriyon, R., Szekely P. and Neches, R. Automatic generation of help from interface design models. *Proceedings of CHI94*, Boston, 1994.

[23] Nass, C., Moon, T., Fogg, B.J., Reeves B. and Dryer, D. C. Can computer personalities be human personalities? *IJ Human-Computer Studies*, 43, 1995.

[24] Novick, D.G. and Tazi, S. Applying the Act-Function-Phase Model to Aviation Documentation. *Proceedings of SIGDOC 98*, Quebec, 1998.

[25] Palanque, P., Bastide, R. and Sengès, V. Validating interactive system design through the verification of formal task and system models. In *Engineering for Human-Computer Interaction*. L J Bass and C Unger (Eds), Chapman and Hall, 1996.

[26] Paris, C. L. The use of explicit User Models in a generation system for tailoring answers to the user's level of expertise. In A Kobsa and W Wahlster (Eds): *User Models in Dialog Systems*. Springer Verlag, 1989.

[27] Paris, C. and Vander Linden K. DRAFTER: an interactive support tool for writing multilingual instructions. *IEEE Computer*, 29, 7, 1996.

[28] Paterno', F. and Mancini C. Developing Adaptable Hypermedia. *Proceedings International Conference on Intelligent User Interfaces*, Los Angeles, January 1999.

[29] Puerta, A.R. and Eisenstein, J. Towards a General Computational Framework for Model-Based Interface Development Systems. *IUI99: International Conference on Intelligent User Interfaces*, Los Angeles, January 1999.

[30] Rickel , J.and Johnson, W. L. Task-oriented dialogs with animated agents in virtual reality. In S Prevost and E Churchill (Eds): *Workshop on Embodied Conversational Characters*, 1998.

[31] Scott, D. Computer support for authoring multilingual software documentation. *ITRI-96-14, Information Technology Research Institute Technical Report Series,* 1996.

[32] Sukaviriya, P., Muthukumarasamy, J., Spaans, A. and de Graaf, H.J.J. Automatic generation of textual, audio and animated help in UIDE. *Proceedings of the Conference on Advanced Visual Interfaces*, Bari, 1994.

[33] Taylor, I.C., McInness, F.R., Love, S., Foster, J.C. and Jack, M.A. Providing animated characters with designated personality profiles. *Proceedings of the Workshop on Embodied Conversational Characters*, S Prevost and E Churchill (Eds), Tahoe City, 1999.

[34] Thimbleby, A. and Ladkin, P. From Logic to manuals. *Proceedings of the IEEE Conference on Software Engineering*, 1997.

[35] Van Biljon, V.R. Extending Petri Nets for specifying man-machine interaction. *International Journal of Man-Machine Studies,* 28, 1988.

[36] Walker, M.A., Cahn, J. E. and Whittaker, S. J. Improvising linguistic style: social and affective bases for agent personality. *Autonomous Agents 97*.