Software Documentation with Animated Agents

Fiorella de Rosis, Berardina De Carolis and Sebastiano Pizzutilo

Department of Informatics, University of Bari, Italy
{derosis,decarolis,pizzutilo}@di.uniba.it
http://aos2.uniba.it:8080/IntInt.html

Abstract

We show 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 we employ is XDM (Context-Sensitive Dialog Modeling), an extension of Coloured Petri Nets that enables representing user-adapted interfaces, simulating their behaviour in different contexts and making semaiutomatic pre-empirical evaluations of consistency and complexity. XDM-Agent is a personality-rich animated character that uses this formal model to illustrate the role of interface objects and to explain which tasks may be performed and how they may be performed. 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, may be adapted to the user characteristics. The potential interest of applying to software documentation the HCI metaphor of 'interacting with a friend' is discussed.

1. Introduction

Documentation is an essential component of software production that requires a considerable investment of resources and time. The result of this effort is often a partially adequate product, that must be revised every time a new release of the application is delivered and/or a new version the documentation, in a different language, has to be prepared. These problems led different research groups to see the production of documentation as strictly linked to software implementation. Documentation may refer to several aspects of software and may be addressed, consequently, to several types of users. It may be aimed at *software understanding*, that is at "the reconstruction of 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 (Erdem et al, 1998); in this case, the main users of documentation produced are software engineers. Alternatively, it may be aimed at producing a *user manual,* that describes how a given application can be used. In this case, the addressees of documentation produced are the end users of the application; the need for information of these users may vary according to the tasks they have to perform, to the frequency of use of the application, to the level of experience with the application itself and so on. The distinction between the two mentioned categories of documentation becomes more fuzzy when the software concerned is a programming language.

The idea of producing a User Manual as a byproduct of interface design and implementation becomes more practicable if a unique 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; the most notable examples in this field are Petri Nets (Palanque et al, 1993) and HUMANOID Hyper Help (Moriyon et al, 1994). In these systems, the 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 (Sukaviriya et al, 1994).

Other projects extended their goal towards the idea of generating, from a knowledge base, the main components of an instruction manual (Thimbleby and Ladkin, 1997; Novick and Tazi, 1998): DRAFTER and ISOLDE are the most significant examples in this field, their main purpose being the generation of multilingual manuals from a unique knowledge base (Paris and Vander Linden, 1996; Scott, 1996; Isolde, Web site). Some of these Projects start from an analysis of the manuals of some wellknown software products (for instance, Mac Write or Tcl-Tk), to examine which types of information these manuals include and which is the linguistic structure of each of them (Hartley and Paris, 1991). 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 that occur during expert consultations, to categorize the users' requests and to identify the strategies they employ for finding the answer to their question issues (Johnson and Erdem, 1991). 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 (Johnson, 1999; André et al, 1998; Lester at al, 1999; Rickel and Johnson, 1998). Some of these Agents combine explanation capabilities with the ability to provide a demonstration of the product, on request; the application fields to which they apply are complex systems or Web sites illustration. Several Projects at Microsoft Research apply this technology to online guide during interaction with a program (Ball et al, 1997).

In this paper, we present the first results of an ongoing Project that is aimed at generating an agent-based online User Manual from a user-adapted model of interaction. After justifying why we selected an Animated Agent as a presentation tool, we describe the formal model we employ (XDM); we then show XDM-Agent's behaviour and how it is influenced by its 'personality'; we give some details about the state of implementation of this system and conclude with some comparison with related works.

2. Why an Animated Agent

The starting point of our research on automated production of software documentation is XDM, a formalism and a tool that we employed in research and teaching for several years, to design, simulate and evaluate user-adapted interfaces (de Rosis et al, 1998). As a first step of our research on software documentation, we studied how this formalism 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 (De Carolis et al, 1998). To this aim, the formal description of interaction was read and interpreted, and schema and ATN-based natural language generation techniques were employed to produce the answers. Shifting from hypertextual helps to an Agent-based manual entails several advantages and raises several methodological problems.

2.1. The advantages

As we said, 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 nonverbal 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, in order to make sure that the user remembers and understands it. A typical example in the software documentation field is deixis: when helps are provided in textual form in a separate window, indicating unambiguously the interface object to which a particular explanation refers is not easy; in HUMANOID, for instance, this problem is solved by employing the same color code to denote the object, in the application and in the help window. An animated agent that can overlap to the application window may solve the same problem much more naturally, 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 behaviour (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, for instance by evoking abstract concepts such as the temporal relations among tasks.

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 (Pelachaud et al, 1999), 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.

2.2 The problems

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 The interface description formalism

To explain how a given application may be used, our Animated Agent employs two knowledge sources:

- a *formal description of the application interface*, in the form of a XDM-model and
- a *description of the strategies that may be employed in generating the explanation*, in the form of a Plan Operators' Library and of a Library of ATNs.

Let us, first of all, describe the first source of knowledge.

XDM (Context-Sensitive Dialog Modeling) is a formalism that extends *Coloured Petri Nets* (CPN: Van Biljon, 1988) 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 description 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 enables the designer to reserve access to particular tasks for particular categories of users and to vary, with the user characteristics, the way tasks are performed and the display layout appears. A detailed description of this formalism may be found in (de Rosis et al, 1998), 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 production of the Animated User Manual, we employ a simplified version of XDM, in which Petri Nets are replaced with UANs (User Action Notations: Palanque et al, 1996). 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'; given two UAN elements, the following relations may hold between them:

- sequence: A° B
- iteration: (A)*
- choice: A | B
- order independence: A & B
- concurrency: A || B
- interruptability: B → A
- interleavability: A ⇔ 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 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; logical and physical projections of these elements 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.

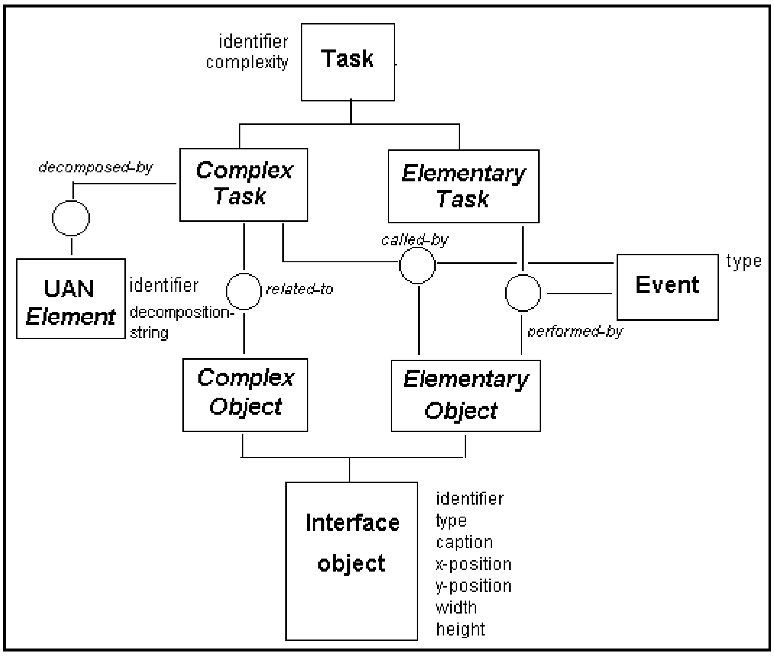

Figure 1: Entity-Relationship representation of the application-KB

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; an elementary task may be performed by a specific Event on a specific elementary Interface Object; an elementary Interface Object may open, as well, 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.

Let us reconsider the example we introduced in Section 3. Let T_i be a UAN element and UAN(T_i) the string that describes how T_i may be decomposed into subtasks, in the UAN language. Let Task(T_i), Obj(T_i) and Ev(Task(T_i), Obj(T_i)) be, respectively, the task associated with T_i, 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): main 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.

4 XDM-Agent's behaviour

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 two-step process; in a planning phase, the presentation content is established; in a realisation phase, the plan is translated into a presentation.The *planning* algorithm thus establishes how the agent will describe the main elements that are related to the window, by reading its description in the application-KB. The presentation plan generated by this algorithm represents how the communicative goal of 'illustrating the window' may be achieved, in the form of a tree; 'primitive subgoals', that cannot be further decomposed, are attached to its leaves. Once the presentation plan for a window is ready, this is given as input to a *realisation* algorithm, that transforms it in a sequence of 'macro-behaviours'. We call 'macro-behaviour' the combination of verbal and nonverbal communicative acts that enables achieving a primitive goal in the plan.

The *list of macro-behaviours* that our agent is able to perform is application-independent but domain-dependent: that is, it is the same for any application to be documented but is tailored to the documentation task. To enable XDM-Agent to describe a user interface, we need to give it the ability to perform the following macrobehaviours:

a. *perform meta-conversational acts*: the agent must 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 macrobehaviour is obtained by combining verbal and nonverbal 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[1] animations, with the aid of a set of auxiliary functions. The list of micro behaviours that are needed in our animated user manual is shown in **Figure 2.** 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.

---

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_Choice
   - 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'

[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.

As we will further discuss in Section 6, the way these micro behaviours are implemented strongly depends on the animations that are included in the software we employ: in particular, a limited overlapping of gestures can be made in MS-Agent, which only enables overlapping speech and text to body animations. We therefore overlap verbal acts to nonverbal ones so that the Agent can move, speech and write something on a ballon at the same time, while we sequence nonverbal acts so as to produce ‘natural’ behaviours. As we mentioned in Section 2, 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 have the role of supporting the communication tasks that could not be effectively achieved with speech (for instance, deixis), of reinforcing concepts that users should not forget (for instance, task relations), of supporting the description of the way actions should be done (for instance, by mimicking events) and, finally, of giving 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’.

5. XDM-Agent’s “personality”

A typical software manual includes three sections[2]:

- a *tutorial*, with exercises for new users,
- a series of step by step *instructions for the major tasks* to be accomplished and
- a ready-reference *summary of commands*.

To follow the ‘minimal manual’ principle (“the smaller the manual, the better”: J M Carrol, cited in Addison and Thimbleby, 1994), 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 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, the agent 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

---

[2] This list of contents was found by Hartley and Paris (1996) in their analysis of a MacWrite manual, and we found the same overall structure in our analysis of the Visual Basic’s hyper-manual.

the Agent, partly of the User), explanations are, initially, much less detailed and a dialog with the User is established, to decide how to go on in the description of the application.
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 (de Rosis and Castelfranchi, 1999):

- 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
- a *literal helper*, that provides a minimal description of the concepts the user explicitly asks to know[3].

These help attitudes may be seen as particular values of the dominant/submissive dimension of interpersonal behaviour[4], which is considered to be the most important factor affecting human-computer interaction (Nass et al, 1995). 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, Nass et al, 1995; but also Dryer, 1996), 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 (Isbister and Nass, 1998; Arafa et al, 1998; Bell and Breese, 1998). By drawing on the cited experiences, we decided to embody the overhelping, dominant attitude in a more 'extroverted' agent that tends to employ a direct and confident phrasing and to gesture and move 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. In order 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[5]: 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.

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 mainly in an object-oriented way in adult encyclopedias, while descriptions in junior encyclopedias tend to be organized in a process-oriented, functional way (Paris, 1989). 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 novices and *Genie* for experts.

---

[3] We give a more detailed definition of these 'dominance' aspects of personality in (de Rosis and Castelfranchi, 1999), where we also define, more in general, the various attitudes towards delegation and help that may be established in multiagent systems.

[4] According to Nass et al, 1995, dominance is marked, in general, by a behaviour that is "self-confident, leading, self-assertive, strong and take-charge"; submissiveness is marked, on the contrary, by a behaviour that is "self-doubting, weak, passive, following and obedient".

[5] Genie and Roby are Microsoft-Agent's 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: minumum 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,and corresponding differences in behaviours

6. 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. In planning, a task-oriented and more detailed or object-oriented and less detailed discourse plan is produced for the two agents. In surface generation, the plan leaves are translated into different realisations of macro-behaviours.

At the planning level, adaptation is made by introducing personality-related conditions in the planning schemas. At the surface realisation level, a unique Behaviour Library is employed in the two cases, with different ways of realising every behaviour. A representation of two portions of plan for Genie and Robby are given in **Figure 4**; this figure shows the difference between a task-oriented and an object-oriented approach to the interface presentation; it shows, as well, that the two plan trees have some leaves in common.
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 as well; if the task 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 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.

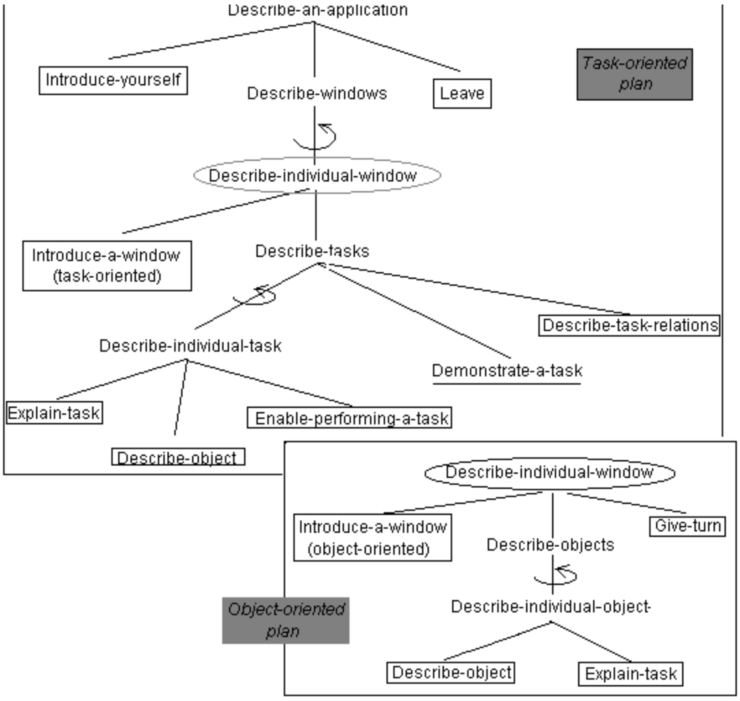

Figure 4: a portion of the task-oriented and of the object-oriented plans

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 5, a** and **b.**

Genie

Macro-Behaviour	***Speech***	***Text balloon***	***Gesture***
Introduce-a-window (task-oriented)	In this window, you can perform the main database management functions:		point-at-area look-at-user
Explain-task Describe-object	- to select a database management function, use the commands in this toolbar;	Select database management functions	move-to-object point-at-object look-at-object look-at-user
idem	- to input identification data, use the textfields in this subwindow;	Input identification data	idem
	....	....	....
Introduce-complex-task	There are two database management functions you may select:		
Explain-task Describe-object Enable-performing-a-task	- to update an existing record, click on the 'Update' button;	Update	move-to-object point-at-object look-at-object look-at-user mimic-click
idem	- to delete a record, click on the 'Delete' button;	Delete	idem
	....	....	....
Describe-task-relation	These tasks are independent of each other; you must choose one of them and perform it.	Choose one	approach-the-user evoke-choice

Figure 5 a: a portion of task-oriented description of a window, by Genie

Robby

Macro-Behaviour	*Speech*	*Text balloon*	*Gesture*
Introduce-a-window (object-oriented)	I'm ready to illustrate you the objects in this window:		look-at-user
Introduce-complex-object	- a toolbar, with 5 buttons:		
Describe-object Explain-task	the first one enables you to update an existing record	Update	point-at-object look-at-object
idem	the second one enables you to delete a record	Delete	idem
	.....		

Figure 5 b: a portion of an object-oriented description of a window, by Robby

7. Interest and limits of our system

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" (Addison and Thimbleby, 1994). An Animated User Manual such as the one we generate is probably suited to the needs of 'novice' users; however, we are not sure that the more complex questions an expert makes can be handled efficiently with our application-KB. We plan to carefully 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 so far (Isbister and Nass, 1999), and we suspect that no general rule can be established in this domain, and that the decision depends on the particular personality traits considered.

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

- the main limit in our generation method is that we do not yet handle *interruptions*: users may take the initiative to make questions to the system only when the agent gives them the turn;
- the second main limit originates from available animations. In MS-Agent, the repertoire of gestures is rather limited, especially considering face and gaze expressions. Some examples: we cannot give an intonation to our Agents' speech; we cannot emphasize the difference in the way they phrase their utterances (a suggestion vs a command) with a difference in 'performative eyes'; we cannot use 'adjectival eyes' to express some concept's property (for instance, a 'difficult' task); we cannot employ 'backchanneling eyes' to provide a feedback to the user, as in the expression of doubt,… and so on [6]. In addition, the difficulty of overlapping animations in MS-Agent 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 (Kendon, 1972), for instance by expressing rhetorical relations among

[6] For a definition of the semantic of the various types of gaze that are employed in human and/or agent conversations, see (Pélachaud et al, 1999).

high-level portions of the plan; an example: a ‘contrast’ between the ways two tasks can be performed. To overcome these limits, we should build our own character, with the mentioned animations.

These limits in our agent’s behaviour having been considered, 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 (with a few exception, as in (André et al, 1998)).

8. Related work

Our research lies in the crossroad of several research 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 behaviour in several contexts and to make pre-emipirical 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 behaviour, 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 the same theories and methods originally developed in the psychological literature for human beings (see, f.i., Nass and colleagues, 1995). 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” (Taylor et al, 1998). Results of the studies by these groups oriented us in the definition of the verbal style and the nonverbal behaviours that characterise XDM-Agent’s personalities (see, f.i., Isbister and Nass, 1999); Walker and colleagues research on factors affecting the linguistic style guided us, as well, in the diversification of the speech attitudes (Walker et al, 1997). Similar factors were applied by other groups, in their definition of their Animated Agents’ personality traits (for instance: Ball and Breese, 1999; Arafa et al, 1999). We contributed, in the past, to the debate about the personality traits that might be relevant in HCI by formalising the cooperation levels and types and the way they may combine, in terms of ‘personality traits’ (Castelfranchi et al, 1998; de Rosis and Castelfranchi, 1999). 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 behaviours. 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.

Acknowledgements

We thank Alessandro Assab and Floriano Cancelli for contributing to implement a prototype of XDM-Agent in the scope of their dissertations in computer science.

References

M Addison and H Thimbleby: Manuals as structured programs. *Proceedings of HCI*, Cambridge University Press, 1994.

E André, T Rist and J Mueller: 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.

J Arafa, P Charlton, A Mamdani and P Fehin: 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.

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

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

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

B De Carolis, F de Rosis and S Pizzutilo: Dynamic generation of on-line help messages from formal dialogue models. DIB-Internal Report, 1998.

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

F de Rosis and C Castelfranchi: 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.

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

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

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

C Isbister and C Nass: Personality in conversational characters: building better digital interaction pertners 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.

Isolde Web site: http://www.syd.dit.csiro.au/staff/cecile/isolde.html

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

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

A Kendon: Some relationships between body motion and speech. In: *Studies in dyadic communication.* A W Siegman and B Pope (Eds), Pergamon, 1972.
J C Lester, J L Voerman, S G Towns and C B Callaway: Deictic believability: coordinated gesture, locomotion and speech in lifelike pedagogical agents. *Applied Artificial Intelligence*, 13 (4-5), 1999.
R Moriyon, P Szekely and R Neches: Automatic generation of help from interface design models. *Proceedings of CHU94*, Boston, 1994.
C Nass, Y Moon, B J Fogg, B Reeves and D C Dryer: Can computer personalities be human personalities? *Int J Human-Computer Studies*, 43, 1995.
D G Novick and S Tazi: Applying the Act-Function-Phase Model to Aviation Documentation. *Proceedings of SIGDOC 98*, Quebec, 1998.
P A Palanque, R Bastide and L Douarte: Contextual help for free with formal dialogue design. *Proceedings of HCI93*, Orlando, 1993.
P Palanque, R Bastide and V Sengès: 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.
C L Paris: 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.
C Paris and K Vander Linden: DRAFTER: an interactive support tool for writing multilingual instructions. *IEEE Computer*, 29, 7, 1996
C Pélachaud, I Poggi and F de Rosis: Study and generation od coordinated linguistic and gaze communicative acts. *Proceedings of AI*IA 99*, 1999.
J Rickel and W L Johnson: Task-oriented dialogs with animated agents in virtual reality. In S Prevost and E Churchill (Eds): *Workshop on Embodied Conversational Characters*, 1998.
D Scott: Computer support for authoring multilingual software documentation. *ITRI-96-14, Information Technology Research Institute Technical Report Series,* 1996.
P Sukaviriya, J Muthukumarasamy, A Spaans and H J J de Graaf: Automatic generation of textual, audio and animated help in UIDE. *Proceedings of the Conference on Advanced Visual Interfaces*, Bari, 1994.
I C Taylor, F R McInness, S Love, J C Foster and M A Jack: Providing animated characters with designated personality profiles. *Proceedings of the Workshop on Embodied Conversational Characters*, S Prevost and E Churchill (Eds), Tahoe City, 1999.
A Thimbleby and P Ladkin: From Logic to manuals. *Proceedings of the IEEE Conference on Software Engineering*, 1997.
V R Van Biljon: Extending Petri Nets for specifying man-machine interaction. *International Journal of Man-Machine Studies,* 28, 1988.
M A Walker, J E Cahn and S J Whittaker. Improvising linguistic style: social and affective bases for agent personality. *Autonomous Agents 97*