Integrating Reactive and Scripted Behaviors in a Life-Like Presentation Agent

Elisabeth André, Thomas Rist and Jochen Müller
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken
Email: {andre,rist,mueller}@dfki.uni-sb.de

1. ABSTRACT

Animated agents - based either on real video, cartoon-style drawings or even model-based 3D graphics - offer great promise for computer-based presentations as they make presentations more lively and appealing and allow for the emulation of conversation styles known from human-human communication. In this paper, we describe a life-like interface agent which presents multimedia material to the user following the directives of a script. The overall behavior of the presentation agent is partly determined by such a script, and partly by the agent's self-behavior. In our approach, the agent's behavior is defined in a declarative specification language. Behavior specifications are used to automatically generate a control module for an agent display system. The first part of the paper describes the generation process which involves AI planning and a two-step compilation. Since the manual creation of presentation scripts is tedious and error-prone, we also address the automated generation of presentation scripts which may be forwarded to the interface agent. The second part of the paper presents an approach for multimedia presentation design which combines hierarchical planning with temporal reasoning.

1.1 Keywords

human-like qualities of synthetic agents, life-like qualities, presentation agents

2. MOTIVATION

A growing number of research projects in both academia and industry provide increasing evidence that the next major step in interface evolution will be a shift towards highly personalized interfaces in which communication between user and computer will be mediated by life-like agents. Due to advances in computer graphics, the realization of visually appealing agents based on real video, cartoon-style drawings or even model-based 3D graphics, has become easier to handle. To be useful, however, such agents have to behave in a reasonable and believable manner.

The focus of our work is on presentation agents appropriate for a broad range of applications including computer-based instruction, guides through information spaces, and web-based product advertisement. There are several reasons for using animated presentation agents in the interface. First, they allow for the emulation of presentation styles common in human-human communication. For example, they enable more natural referential acts that involve locomotive, gestural and speech behaviors (cf. [9]). In virtual environments, animated agents may help users learn to perform procedural tasks by demonstrating their execution (cf. [12]). Furthermore, they can also serve as a guide through a presentation to release the user from orientation and navigation problems common in multi-window/multi-screen settings (cf. [3]). Last but not least, there is the entertaining and emotional function of such animated characters. They may help to lower the "getting started barrier" for novice users of computers/applications, and, as Adelson notes, "... interface agents can be valuable educational aids since they can engage students without distracting or distancing them from the learning experience" (cf. [1], pp. 355).

To illustrate this, we use some examples taken from the PPP (Personalized Plan-based Presenter) system. The first application scenario deals with instructions for the maintenance and repair of technical devices, such as modems. Suppose the system is requested to explain the internal parts of a modem. One strategy is to generate a picture showing the modem's circuit board and to introduce the names of the depicted objects. Unlike conventional static graphics where the naming is usually done by drawing text labels onto the graphics (often in combination with arrows pointing from the

label to the object), the PPP Persona enables the emulation of referential acts that also occur in personal human-human communication. In the example, it points to the transformer and utters "This is the transformer" (using a speech synthesizer). The example also demonstrates how facial displays and head movements help to restrict the visual focus. By having the Persona look into the direction of the target object, the user's attention is directed to this object.

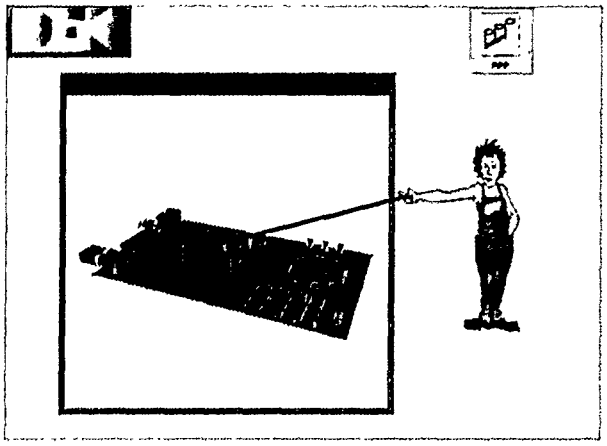

Figure 1: The Persona instructs the user in operating a technical device

In the second example, the Persona advertises accommodation offers found on the WWW. Suppose the user is planning to spend holidays in Finland and is therefore looking for a lake-side cottage. To comply with the user's request, the system retrieves a matching offer from the web and creates a presentation script for the PPP persona which is then sent to the presentation viewer (e.g. Netscape Navigator^{TM} including a Java^{TM} interpreter). When viewing the presentation, the PPP Persona highlights the fact that the cottage has a nice terrace by means of a *verbal annotation of a picture*; i.e., Persona points to the picture during a verbal utterance (cf. Fig. 2). In case graphics is generated automatically, as in the modem example, the presentation system can build up a reference table that stores the correspondence between picture parts and domain concepts. Since scanned pictures are used in the travelling agent application, such a reference table has been set up manually in order to enable pointing gestures to that material. However, in many cases, the author of a web page already did the job of relating image regions to concepts. For example, many maps available on the web are already mouse-sensitive and the system just has to follow the links to find the concepts related to the mouse-sensitive regions.

There are several requirements an animated presentation agent has to meet. According to its functional roles in a presentation, the animated character must be conversant with a broad variety of presentation gestures and rhetorical body postures. Furthermore, it should adopt a reasonable and lively behavior without being distracting. From the technical point of view, the declarative specification of agent behaviors should be supported, and the software that realizes the *character player* should be highly independent of application and platform.

The overall behavior of the PPP persona is partly determined by a presentation script, and partly by the agent's self-behavior. In the next section, we introduce a declarative specification language for defining such behaviors. Behavior specifications are used to automatically generate a control module of a presentation display component. This display component will be referred to as the Persona Engine throughout the rest of the paper. When operated stand-alone, the Persona Engine expects a presentation script as input and displays the corresponding presentation as output. However, for an increasing number of applications, the *manual* authoring of such presentation scripts (including the creation of the media material such as text, graphics, and animation clips) is no longer feasible because information has to be communicated fast and flexibly to meet the specific needs of the individual presentation consumer. Although powerful authoring tools have become available, they are not likely to solve this problem as they only facilitate presentation editing. Therefore, we also address the automatization of the whole authoring process, including the selection of appropriate information content, media allocation, logical presentation structuring, creation of media items which are to convey selected information as well as the temporal scheduling of presentation acts.

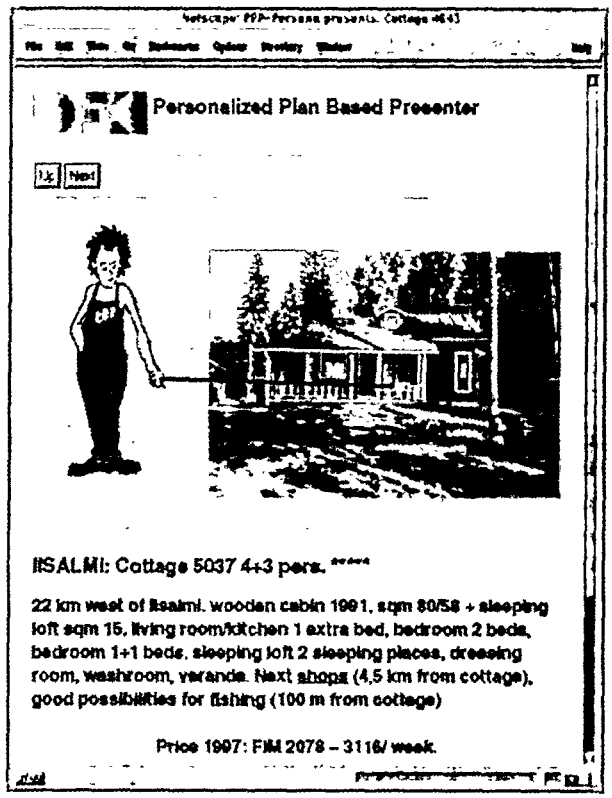

Figure 2: PPP Persona presents retrieval results from the web using the Netscape Navigator^{TM} and Java^{TM}

3. CONCEPTION OF THE PERSONA ENGINE

As stated above, the Persona Engine takes presentation scripts as input and displays the corresponding presentation. However, in contrast to other display components for media objects (e.g. video or audio players, graphics viewers, etc.) the output of the Persona Enginge is not only determined by the directives (i.e., presentation tasks) specified in the script. Rather, the behavior of the animated character follows the equation:

Persona behavior := directives + self-behavior

Such self-behaviors are indispensible in order to increase the Persona's vividness and believability. Currently, they comprise idle-time actions, such as tapping with a foot, actions for indicating activity, e.g., turning over book pages, navigation acts, such as walking or jumping, and immediate reactions to external events, such as mouse gestures on the presented material.

3.1 Defining the Persona's Behaviors

To facilitate the definition of Persona behaviors, we have developed a declarative specification language. Persona behaviors are represented as operators of a planning system relying on standard representation constructs common in AI planning (e.g., see [2]). For instance, the following definition specifies the pre- and postconditions for the action: *bottomupjumping*.

```
(defprimitive bottomupjumping
    :pre ((leftarm standard)(rightarm standard)
          (icon noicon)(bodydir front)
          (bodypos stand)(stick off))
    :post ((posy -= 1))
    :gesture 42)
```

The action can only be performed if both arms are in a standard position, the Persona is not iconified, is facing the user, is standing, and isn't holding a stick. If this is the case, the image sequence associated with the action (*:gesture 42*) is played and the state of the world is updated as indicated in the post conditions *:post*. Otherwise, the system tries to achieve the desired preconditions by executing further actions.

While primitive actions like *bottomupjumping* are directly associated with an image sequence, complex actions are composed of several subactions. An example of a complex action is:

```
(defactionseq MoveUp
    :pre ((icon noicon)(bodydir front)
          (leftarm standard)(rightarm standard)
          (bodypos stand)(stick off))
    :prim startbottomupjumping
    :while ((posy ≠ target) :prim bottomupjumping)
    :prim endbottomupjumping)
```

This definition specifies a jump to a given target *target*. The preconditions of this action coincide with the preconditions of *bottomupjumping*. If they are satisfied, the Persona starts with the jump (*startbottomupjumping*) and continues to jump until it reaches the target (*(posy ≠ target)*). Finally, it finishes the jump (*endbottomupjumping*).

3.2 Compiling Behaviors

Since animations have to be performed in realtime (and our system should run on ordinary PCs/workstations as well), it's not advisable to decompose actions into animation sequences at runtime. Following [7], we developed a multi-pass compiler that enables the automated generation of a finite state machine from declarative action specifications. The state machine is then converted into efficient machine code (cf. Fig. 3). That is we compute for all possible situations beforehand which animation sequence to play. As a result, the system just has to follow the paths of the state machine when making a decision at runtime.

When creating the source code for the finite state machine [1], action specifications are translated as follows:

- *Primitive Actions*, such as *bottomupjumping*, are mapped onto functions in which (1) a function is called to achieve the precondition of the action, (2) a command for playing an image sequence is executed and (3) the settings of the state variables are updated.
- *Complex Actions*, such as *MoveUp*, are mapped onto functions which may invoke other functions according their control structures. For example, the middle part of Fig. 3 lists the source code for the action *MoveUp*.
- *Idle-Time Actions* are mapped onto functions which apply heuristics to select an idle-time script, play it and update the state variables.

The next step is the definition of functions for achieving the preconditions specified in the action definitions. In particular, we have to compute action sequences which transform each possible state in which the system might be into these preconditions. This is done by regression-based planning. For each precondition specified in an action definition, we apply primitive actions in reverse order until all possible start states are achieved. The result of this process is a list of tuples that consist of a state and an action sequence which has to be performed in this state to satisfy the precondition. These tuples are converted to *if-then-else* programme blocks and compiled into efficient machine code.

The compiled state machine forms the so-called *behavior monitor*. Summing up, the behavior monitor accomplishes the following tasks:

- It decomposes complex presentation tasks into elementary postures that correspond to a single Persona frame (e.g. stored as a pixmap) or to an uninterruptable image sequence.

[1] We are able to generate both C- and Java-Code.

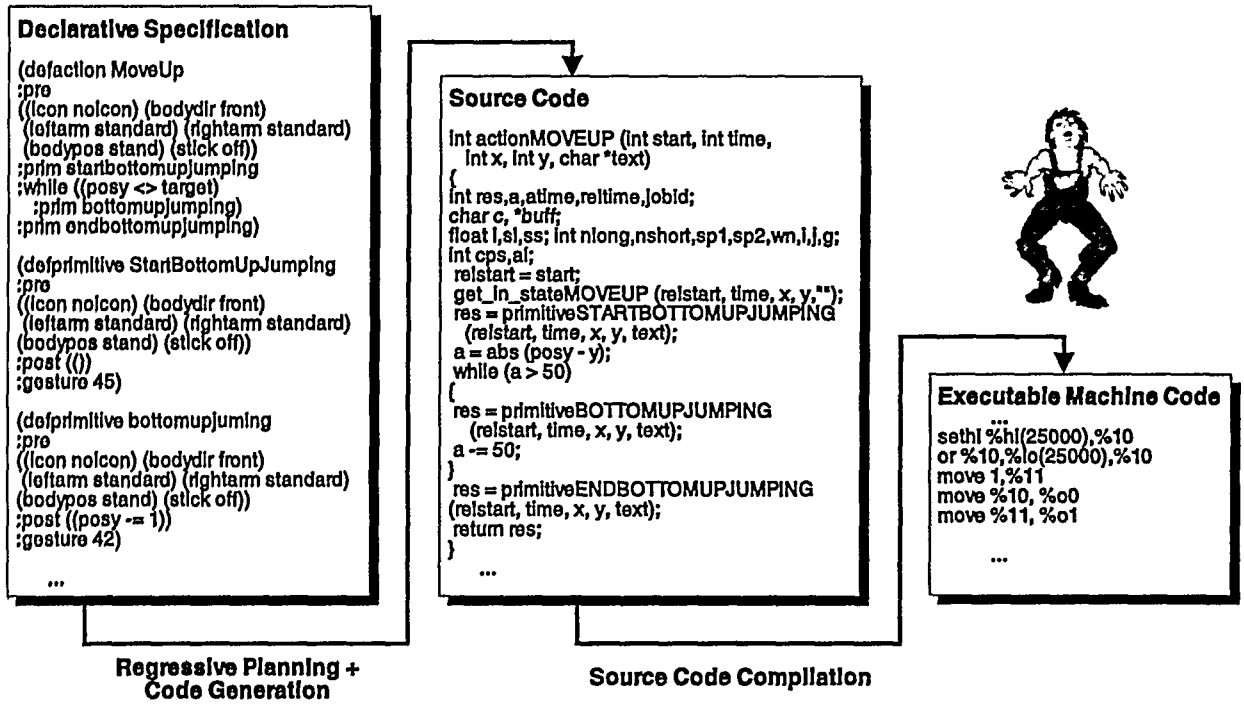

Figure 3: Compilation of Persona Actions

- It ensures that necessary preconditions for the execution of primitive actions are satisfied.
- It updates the internal state of the Persona after the execution of a primitive posture.
- It augments the Persona's presentation behaviors by believability-enhancing behaviors, such as idle-time acts.

Besides the *behavior monitor*, the Persona Engine also comprises an *event handler*, a *character composer*, and an *interface* which is tailored to the target platform (currently either X11 or a Java^{TM} interpreter). The task of the *event handler* is to recognize whether input derived from the platform interface needs immediate responses from the Persona. That is, for each input message the event handler checks whether the message triggers one of the so-called "reactive behaviors" stored in an internal knowledge-base. If this is the case, the selected behavior is made accessible to the behavior monitor. Depending on the application, notifications may be forwarded to the application program, too. For example in our PPP system, some events are interpreted as requests for the satisfaction of new presentation goals and thus activate a presentation planner (thus the dotted line in Figure 4). The postures determined by the behavior monitor are forwarded to a *character composer* which selects the corresponding frames (video frames or drawn images) from an indexed data-base, and forwards the display commands to the window system.

4. PLANNING PRESENTATION SCRIPTS

For many applications, the manual specification of presentation scripts is not feasible. This encouraged us to address the automatic creation of presentation scripts.

To build up a coherent and temporally coordinated presentation for a specified presentation goal, the PPP system exploits design knowledge. In our approach, we use so-called presentation strategies to represent that knowledge. They are characterized by a *header*, a set of *applicability conditions*, a collection of *inferior acts*, a list of *qualitative* and *metric* temporal constraints and a *start* and an *end interval*. The header corresponds to a complex presentation act. The applicability conditions specify when a strategy may be used and constrain the variables to be instantiated. The inferior acts provide a decomposition of the header into more elementary acquisition or presentation acts. While acquisition acts refer to the retrieval or creation of multimedia material, presentation acts refer to its display.

Currently, our Persona is able to perform gestures that: express emotions (e.g., approval or disapproval), convey the communicative function of a presentation act (e.g., warn, recommend or dissuade), support referential acts (e.g., to look at an object and point at it), regulate the interaction between the Persona and the user (e.g., establishing eye contact with the user during communication) and indicate that the Persona is speaking. Of course, these gestures may also superimpose each other. For example, to warn the user, the Persona lifts its index finger, looks towards the user and utters the warning.

Information concerning the temporal coordination of presentation acts is represented by means of *qualitative* and *quantitative constraints*. Qualitative constraints are represented in an "Allen-style" fashion which allows for the specification of thirteen temporal relationships between two named intervals, e.g. *(Speak1 (During) Point2)*. Quantitative constraints appear as metric (in)equalities, e.g. *(5 ≤ Duration Point2)*. For more details, see [5]. An example of a presentation strategy is listed below. It may be used to build up the presentation shown in Fig. 1.

```
(defstrategy
    :header (Introduce Persona User ?object ?window)
    :applicability-conditions
    (Bel Persona (ISA ?object Physical-Object))
    :inferiors
    ((A1 (Make-Window Persona User ?object ?window))
     (A2 (S-Show-Window Persona User ?window ?object))
     (A3 (Elaborate-Parts Persona User ?object ?window))
     (A4 (S-Wait Persona User)))
    :qualitative: ((A1 (meets) A2) (A3 (starts) A2)
                   (A3 (meets) A4)(A4 (finishes) A2))
    :metric ((10 ≤ Duration A2) (2 ≤ Duration A4 ≤ 2))
    :start A1
    :finish A2)
```

Besides acts for the acquistion of multimedia material, such as *Make-Window*, our strategies also comprise presentation acts to be executed by the Persona, such *S-Show-Window*. Note that we are not forced to completely specify the temporal behavior of all acts at definition time. This enables us to handle acts with *unpredictable* durations, start and endpoints, i.e. acts whose temporal behavior can only be determined by executing them. For example, in the strategy we only specify a minimal duration for act A2 and a fixed duration for act A4.

In order to construct presentation scripts, we have combined a hierarchical planner (cf. [4]) with a temporal reasoner which is based on MATS (Metric/Allen Time System, cf. [8]). The basic idea behind the planning process is as follows: Given a presentation goal, try to find a matching strategy and post the inferior acts of this strategy as new subgoals. For each subgoal, create a local temporal constraint network which contains all qualitative and metric constraints corresponding to the applied strategy. In case a subgoal cannot be achieved or the temporal constraint network proves inconsistent, apply another matching strategy. The goal refinement process terminates if all goals are expanded to elementary acquisition or presentation acts. To allow for user interaction, some goals are realized as mouse-sensitive items in the final presentation and only expanded on demand, i.e., if the user clicks on the corresponding item at presentation runtime. After the completion of the presentation planning process, PPP determines the transitive closure over all qualitative constraints and computes numeric ranges over interval endpoints and their difference. The last step is the creation of a schedule which reflects the temporal behavior of the presentation. Since the behavior of many events is not predictable, the schedule still permits time to be stretched or shrunk between them. At runtime, the initial schedule is refined by adding new metric constraints to the constraint network. Fig. 5 provides an overview of the presentation planning process.

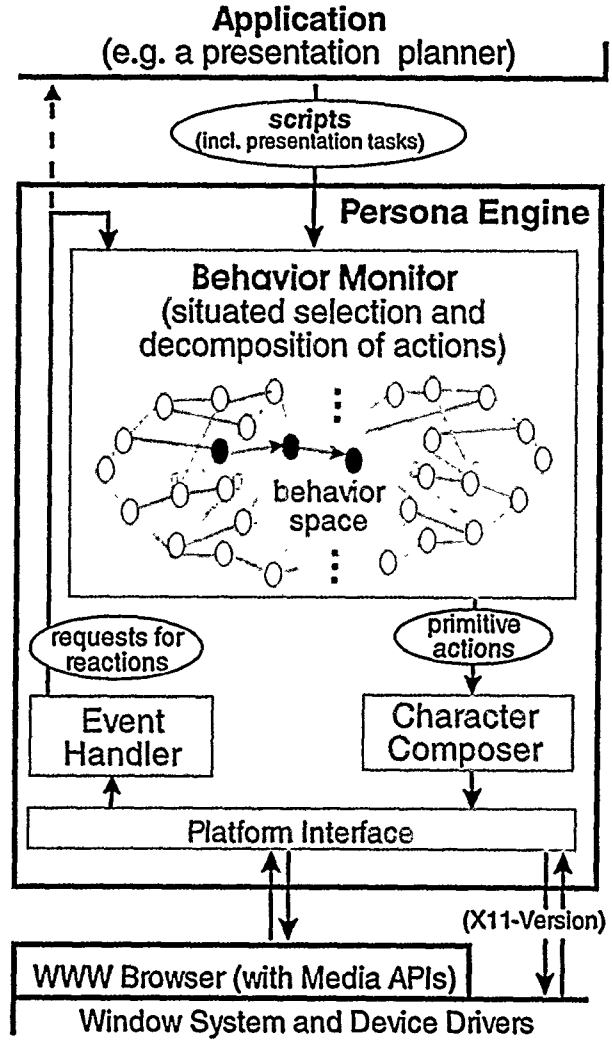

Figure 4: Architecture of the Persona Engine

5. EVALUATION

Our research on animated interface agents was motivated by the assumption that they make man-machine communication more effective. In order to find empirical support for this conjecture, we conducted a psychological experiment in which 28 subjects (15 females, 13 males, average age: 28) were confronted with 5 web-based presentations that they were subsequently asked questions about. All presentation scripts were created using the authoring tool described in this paper. The subjects were allowed to spend as much time as they required to answer the questions, but not to watch the presentations several times. On the average, each subject spent 45 minutes on the experiment.

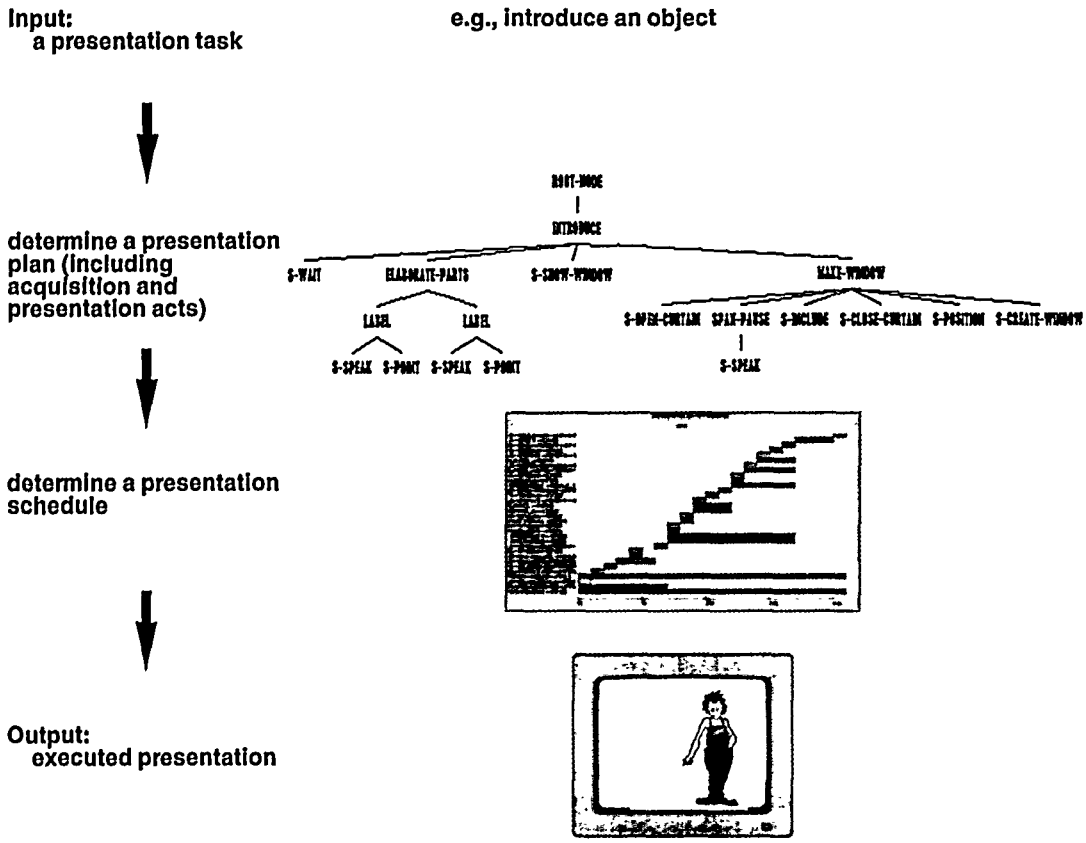

Figure 5: The Presentation Planning Process

Our study focused on two issues: 1) the effect of a Persona on the subjects' rating of the presentations (a subjective measure), and 2) its effect on the subjects' comprehension of presentations (an objective measure). The first issue was measured through a questionnaire at the end of the experiment. The second issue was measured through comprehension and recall questions following the presentations.

In the experiment, two variables were varied. The first variable concerned the Persona itself. The Persona was either absent or present. In the experiment without the Persona, a voice spoke the same explanations as in the Persona-version and pointing gestures by the Persona were replaced with an arrow. The second variable was the information type. Subjects were confronted with technical descriptions of pulley systems and with person descriptions (i.e., information about DFKI employees). The first variable was manipulated between-subjects, while the second variable was manipulated within-subjects. Thus, each subject viewed either presentations with or without the Persona, but each subject was confronted with both kinds of presentation.

Concerning our first objective, the evaluation of the Persona's affective impact, our study revealed a positive effect. Most subjects perceived the Persona as being helpful. Only one subject indicated that he would prefer presentations without a Persona in the future. Furthermore, subjects confronted with the Persona-based presentations rated the technical descriptions as less difficult and more entertaining. In the case of the DFKI experiment, we didn't find a significant difference between the ratings of the difficulty of the presentation and its entertaining value. Also subjects found the Persona's behavior less adequate in this domain. We hypothesize that this result is due to the fact that the Persona's realization as a workman is more appropriate to technical descriptions than to institute descriptions.

Concerning the second objective, the evaluation of the Persona's learning effect, the difference between the Persona and the No-Persona version was not statistically significant. That is, the Persona did neither contribute to the students' comprehension of the technical matters in the pully experiment, nor to their recall capabilities in the second experiment. As a possible reason, we indicate that we only exploited Persona behaviors that can be easily replaced with other means of communication not necessarily requiring the existence of a Persona. In our experiments, Persona gestures were restricted to neutral facial expressions (i.e. head and eye movements towards the objects currently being explained and lip movements indicating that the Persona is speaking), pointing gestures and simple idle time actions, such as breathing or tapping with a foot.

On the other hand, initial concerns that people would be dis-

tracted by the Persona and concentrate too much on the Persona's facial expressions instead of looking at the referent of the pointing gestures were not confirmed. In the questionnaire, all subjects indicated that the Persona did not distract them.

6. RELATED WORK

In this section, we will review previous approaches for authoring and controlling the behavior of life-like characters.

Closely related to our work is Microsoft's Persona project in which the interface agent is a parrot named Peedy (cf. [7]). Nevertheless Peedy is an anthropomorphic character since it interacts with the user in a natural-language dialogue, and also mimics some non-verbal (human) communicative acts, e.g., Peedy raises a wing to the ear in case speech recognition fails. Since Peedy is to act as a conversational assistant (at least for the sample application, a retrieval system for music CD's), the system comprises of components for processing spoken language, dialogue management and the generation of audio-visual output. However, the system doesn't have to create presentation scripts since the presentation of material is restricted to playing the selected CDs.

Lester and Stone [13] have combined a coherence-based behavior sequencing engine to control the behavior of *Herman the Bug*, the pedagogical agent of *Design a Plant*. This engine dynamically selects and assembles behaviors from a behavior space consisting of animated segments and audio clips. This material has been manually designed by a multidisciplinary team of graphic artists, animators, musicians and voice specialists. On the one hand, this allows the authoring of high quality presentations as the human author has much control over the material to be presented. On the other hand, enormous effort by the human author is required to produce the basic repertoire of a course. In contrast to their work, our approach aims at a higher degree of automatization. The basic animation units from which a presentation is built correspond to very elementary actions, such as taking a step or lifting one's arm, which are flexibly combined by the Persona Engine. Furthermore, we don't rely on prestored audio clips, but use a speech synthesizer to produce verbal output.

Rickel and Johnson [12] have developed a pedagogical agent called Steve based on the Jack Software, a tool for modeling 3D virtual humans [6]. Instead of creating animation sequences for a course offline and putting them dynamically together as in *Design a Plant*, the 3D character Steve is directly controlled by commands, such as "look at", walk to" or "grasp an object". In this case, the character interacts with virtual objects in the same way as a human will do in a real environment with direct access to the objects. In contrast to this, our system strictly distinguish between domain and presentation objects. That is the PPP Persona is part of a multimedia presentation and interacts with domain objects via their depictions or descriptions. This setting is similar to a setting where a tutor presents and comments slides or transparencies.

Similar applications have been described by Noma and Badler [10] who developed a virtual human-like presenter based on the Jack Software and Thalmann and Kalra [14] who produced some animation sequences for a virtual character acting as a television presenter. While the production of animation sequences for the TV presenter requires a lot of manual effort, the Jack presenter receives input at a higher level of abstraction. Essentially, this input consists of text to be uttered by the presenter and commands, such as *pointing* and *rejecting*, which refer to the presenter's body language. Nevertheless, the human author still has to specifiy the presentation script, while our system computes this automatically starting from a complex presentation goal. However, since our presentation planner is application-independent, it may also be used to generate presentation scripts for the Jack presenter or the TV presenter.

Perlin and Goldberg [11] have developed an "english-style" scripting language called IMPROV for authoring the behavior of animated actors. To a certain extent, the library of agent scripts in their approach can be compared to the repertoire of presentation strategies in our approach since they both allow for the organization of behaviors into groups. However, their scripts are represented as a sequence of actions or other scripts while we exploit the full set of Allen relationships. A novelty of our system is that it doesn't require the human author to specify the desired temporal constraints between the single presentation acts, but computes this information dynamically from a complex presentation goal. Furthermore, our system does not only design presentation scripts, but also assembles the multimedia material to be presented to the user.

7. TECHNICAL DATA

Implementations of the PPP Persona Engine are currently available for Unix platforms running X11, and Java-enhanced WWW-browsers. The Persona Engine has been implemented in Java^{TM} and C++. It relies on about 250 frames for each Persona. Currently, we use two cartoon personas and three real personas composed of grabbed video material. To control the behavior of the personas, more than 150 different behaviors have been defined. The presentation planner, the temporal reasoner and the Persona Compiler have been implemented in Allegro Common Lisp. To plan presentation scripts, about 70 presentation strategies have been defined.

8. CONCLUSION

Animated user interface agents have been proposed by several other authors. Distinguishing features of our approach are:

- *the clear distinction between task-specific directives and character- and situation-specific self-behaviors*

Such a distinction has several advantages. From a conceptual point of view, it's more adequate to draw a clear borderline between a *"what to present"- part* which is determined by the application, and a *"how to present"-part* which also depends on the particular presenter and the current situation. From the practical perspective, the approach facilitates script generation since scripts can be formulated on a higher level of abstraction. The distinction is also reflected by different processing mechanisms. While the design of a presentation script is performed in a proactive planning phase, the transformation of these scripts into fine-grained animation sequences is done reactively taking into account the presentation situation at runtime.

- *the integration of a temporal reasoner*

  Most commercial systems require the human author to completely specify the temporal behavior of a multimedia presentation by positioning events along a time line, an error-prone and tedious process. More sophisticated scripting approaches allow for the specification of a presentation at a higher degree of abstraction. But, the human author still has to input all desired temporal constraints from which a consistent schedule is computed. In our approach, schedules are generated automatically starting from a complex presentation goal. Furthermore, our system ensures temporal consistency at presentation runtime by continuously adapting schedules whenever necessary. This has been achieved by combining an AI planning approach with a module for temporal reasoning.

While our evaluation study did not support the assumption that life-like agents improve task comprehension and information recall cababilities of human presentation consumers, it clearly revealed a strong affective impact. Our subjects rated learning tasks presented by the Persona as less difficult than presentations without a life-like character. Obviously however, this effect does not occur in all applications, and users seem to have clear preferences about when to have a personified agent in the interface. Thus, user interface designers should not only take into account inter-individual, but also intra-individual differences.

9. ACKNOWLEDGMENTS

This work has been supported by the BMBF under the contracts ITW 9400 7 and 9701 0. We would like to thank Peter Rist for drawing the cartoons, H.-J. Profitlich and M. Metzler for the development of the temporal reasoner, Frank Biringer for implementing the Persona Compiler, and Susanne van Mulken for supervising the empirical evaluation.

10. REFERENCES

1 B. Adelson. Evocative Agents and Multi-Media Interface Design. In *Proc. of the UIST'92 (ACM SIGGRAPH Symp. on User Interface Software and Technology)*, pages 351–356, Monterey, CA, U.S.A., 1992.

2 J. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning*. Morgan Kaufmann, San Mateo, California, 1990.

3 E. André, J. Müller, and T. Rist. WebPersona: A Life-Like Presentation Agent for the World-Wide Web. In *Proc. of the IJCAI-97 Workshop on Animated Interface Agents: Making them Intelligent*, Nagoya, 1997.

4 E. André and T. Rist. The Design of Illustrated Documents as a Planning Task. In M. Maybury, editor, *Intelligent Multimedia Interfaces*, pages 94–116. AAAI Press, 1993.

5 E. André and T. Rist. Coping with temporal constraints in multimedia presentation planning. In *Proc. of AAAI-96*, volume 1, pages 142–147, Portland, Oregon, 1996.

6 N.I. Badler, C.B. Phillips, and B.L. Webber. *Simulating Humans: Computer Graphics, Animation and Control*. Oxford University Press, New York, Oxford, 1993.

7 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 at microsoft. In J.M. Bradshaw, editor, *Software Agents*. AAAI/MIT Press, Menlo Park, CA, 1997.

8 H. A. Kautz and P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. of AAAI-91*, pages 241–246, 1991.

9 J. Lester, J.L. Voerman, S.G. Towns, and C.B. Callaway. Cosmo: A life-like animated pedagogical agent with deictic believability. In E. André, editor, *Proc. of the IJCAI-97 Workshop on Animated Interface Agents: Making them Intelligent*, Nagoya, 1997.

10 T. Noma and N.I. Badler. A Virtual Human Presenter. In *Proc. of the IJCAI-97 Workshop on Animated Interface Agents: Making them Intelligent*, pages 45–51, Nagoya, 1997.

11 K. Perlin and A. Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Computer Graphics*, 28(3), 1996.

12 J. Rickel and W.L. Johnson. Integrating pedagogical capabilities in a virtual environment agent. In *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, 1997.

13 B.A. Stone and J.C. Lester. Dynamically sequencing an animated pedagogical agent. In *Proc. of AAAI-96*, volume 1, pages 424–431, Portland, Oregon, 1996.

14 N. Magnenat Thalmann and P. Kalra. The Simulation of a Virtual TV presenter. In *Computer Graphics and Applications*, pages 9–21. World Scientific, 1995.