An Agent-based Architecture for Distributed Interfaces and Timed Media in a Storytelling Application

Jun Hu
Media Interaction Group
Philips Research Laboratories
Prof. Holstlaan 4,
5656AA Eindhoven, The Netherlands
hu@natlab.research.philips.com

Loe Feijs
Department of Industrial Design
Eindhoven University of Technology
Postbus 513
5600MB Eindhoven, The Netherlands
l.m.g.feijs@tue.nl

ABSTRACT

This paper presents an agent-based architecture that enables the presentation of interactive timed media to distributed and networked devices. The architecture provides an automatic mapping of the same document to different environments. It uses a hierarchy of PAC (Presentation-Abstraction-Control) agents, which allows separating the presentation aspects from the control aspects and dynamically registering the new PAC agents to the system.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented and virtual realities*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, multiagent systems*

General Terms

Design, Experimentation, Human Factors

Keywords

Agent-based architecture, distributed interfaces, timed media

1. INTRODUCTION

In Philips' vision of Ambient Intelligence [1] the next age of people's interactive media experience will not unfold only on a computer or television, or in a headset, but in a whole physical environment which is responsive to people's movements, gestures, speech and touch, and by nature, these environments differ from each other. To deliver interactive media to such environments, we need an adaptive system architecture to map the media to personalized environment configurations.

Ambient Intelligence (AmI) is firstly artificial intelligence (AI). An AmI system can be viewed as a network of interface agents and supporting agents. Interface agents, later defined as Interactors in this paper, have their physical embodiments, the sensors and effectors of which talk directly to the external world or the end users. Multimedia content can be distributed to multiple interactors to increase the immersive media experiences.

The carrier for this work is an interactive storytelling application (TOONS) for children in the NexTV project sponsored by the European Commission under the IST-programme. The TOONS presents the interactive show in an environment which includes a big screen, a robot and a light. According to the user requirements, the system architecture has to support: (1) Distributed interfaces require that not only the content presentation, but also the user input and control are distributed over the networked environment. (2) The story will be presented to different user environments which configurations may change during the time. (3) Not only the media, but also the user interactions are timed and should be synchronized.

2. PRESENTATION SPECIFICATION

Firstly we need a model for scripting the interactive media presentation. The unknown configuration of environments requires the presentation to be described at a high level of abstraction.

We model the interactive media presentation as an interactive *Story* presented in a desired environment (called a *Theater*). The story consists of several *Storylines* and a definition of possible user *Interaction* during the story. User interaction can result in switching between storylines, or changes within a storyline. *Dialogues* make up the interaction. A dialogue is a linear conversation between the system and the user, which in turn consists of *Feed-forward* content, and the *Feedback* content depending on the user's *Response*. The environment may have several *Interactors*. The Interactors render the media objects. And finally, the story is rendered in a *Theater*.

Interactors are interface agents that are present in an interactive presentation as actors and assistants. An Interactor is a self-contained autonomous agent which has an expertise of data processing and user interaction. It is able to abstract the user inputs as events and to communicate with other Interactors. An Interactor can be present in an environment as a software entity, alive in a computer system or embodied in a hardware device. Storylines, feed-forward and feedback components can be recognized as time-based media objects. A media object can be rather abstract, for example, expressions, behaviors, and even emotions, can be defined as media objects as long as they can be recognized and rendered by any Interactors.

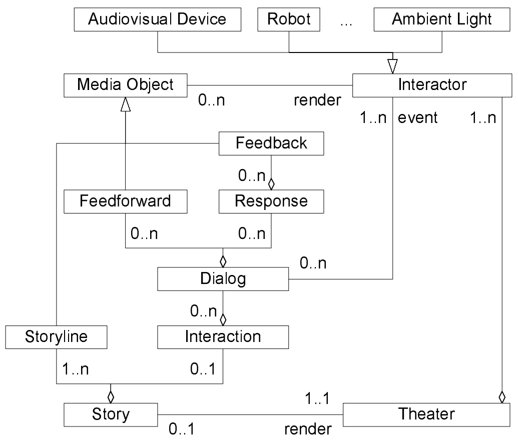

Figure 1: Object oriented model of a presentation

A Story is scripted in StoryML. StoryML is an XML based specification language for interactive stories which aims at a technical solution for the interactive and distributed media representation. Roughly speaking, the StoryML specification is a just script containing references to various media objects, next to the *Theater* requirements and the interaction definitions, telling how to turn them into a single presentation. The StoryML reflects directly many concepts from an object-oriented model (Figure 1). It provides a higher level of abstraction which is independent of media representation technologies. The detailed definition and an example script can be found in [3].

3. AGENT-BASED RENDERING SYSTEM

For an adaptive architecture, we need a structural framework to separate the user interface from the application. Many agent-based models have been developed along the lines of the object-oriented paradigms. MVC (Model-View-Controller) and PAC (Presentation-Abstraction-Control) are the most popular and often used ones. For presenting media onto distributed interfaces, we prefer the PAC to the MVC, because the PAC agents are more self-contained, and have a dedicated controller for communications. The PAC is more suitable for a distributed environment [2].

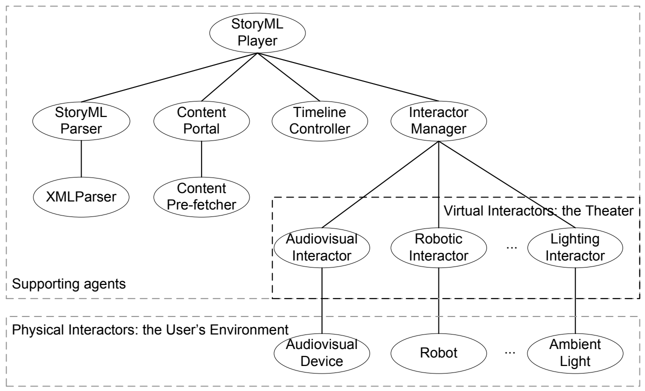

Figure 2: Architecture of the rendering system

Figure 2 shows the PAC-based hierarchical structure of the rendering system. The content portal establishes the connection to content servers and provides the system with timed content. The content pre-fetcher overcomes the start latency by pre-fetching a certain amount of data and ensures that the media objects are prepared to start at specified moments. An XML parser first parses the StoryML document into Document Object Model (DOM) objects and then the StoryML parser translates the DOM objects into internal representations. The Timeline controller acts as a timing engine that plays an important role in media and interaction synchronization. The bottom level agents indicate different physical interactors. For each physical agent, there is an intermediate virtual interactor connected as its software counterpart. All virtual interactors are managed by an Interactor manager.

The mapping from the desired Theater to the user's physical environment is done autonomously and dynamically by the virtual interactors without the upper layer manager intervention. A virtual interactor first finds out if there is a physical device available that can be a replacement. This is done by broadcasting a query to the network and every networked device will return a token together with a description of its capability. If found, the media presentation and interaction tasks are then transferred to the device identified by its token. The virtual interactor maintains the connection with the physical one by sending handshaking messages on a regular basis. Once no response is from the physical device, the virtual interactor tries to find another one in the environment. If this failed, the virtual interactor presents itself onto an audiovisual device in order to make the media presentation and the user interaction still possible.

The top layer player agent analyzes the StoryML document to build a schedule for the presentation, which is managed by the timeline controller. The synchronization is then managed by the timeline controller globally along the hierarchy, and carried out by the content pre-fetcher and interactors locally.

4. CONCLUSIONS

The agents in the architecture are autonomous, social, reactive and pro-active [4]. The agents are autonomous, as shown by the interactors, which take care of complex user interactions without manager intervention. The agents have social ability since they communicate with other agents using the hierarchical control network. The agents are reactive by adapting to the availability of physical interactors. The agents are pro-active by working on the goal and derived sub-goals of putting a given StoryML description into reality in a dynamic environment.

5. REFERENCES

[1] E. Aarts, R. Harwig, and M. Schuurmans. *Ambient intelligence*, pages 235–250. McGraw-Hill, Inc., 2001.

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, A System of Patterns*. John Wiley & Sons, Inc., 1996.

[3] J. Hu. *Distributed Interfaces for a Time-based Media Application*. Post-master thesis, Eindhoven University of Technology, 2001.

[4] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.