Collaborative T-learning: Bringing Greater Levels of Interactivity into the Home

Andrés Elexpuru Eguia, Martín López Nores, Yolanda Blanco Fernández, José Pazos Arias, Belén Barragáns Martínez, Jorge García Duque, Alberto Gil Solla, Manuel Ramos Cabrer
Department of Telematic Engineering, University of Vigo, 36310, Spain
{elexpuru, mlnores, yolanda, jose, belen, jgd, agil, mramos}@det.uvigo.es

Abstract

T-learning —the provision of educational services over Interactive Digital TV— is emerging as an important medium to create opportunities for learning at home. In this paper, we elaborate on a framework for the development and deployment of collaborative t-learning services, presenting an architecture that allows distributing the logic of a service, a way to define complex interaction patterns among users and a network infrastructure that provides adequate support for the communication needs.

1. Introduction

As a platform for education, Interactive Digital TV is considered a key to reach the widest audiences, since it meets the socially important need to offer online learning services to people who cannot afford to buy a computer, do not have Internet access or lack the knowledge to use such technologies. Thus, t-learning —the provision of educational services over IDTV— is emerging as a new way to create opportunities for learning at home [1, 8].

So far, the t-learning offer has been mostly based on broadcast contents —services with one-way communication, from service providers to users—, but it is commonly agreed (see [2]) that the prospects will be enhanced by supporting greater levels of interaction between users and service providers. Unfortunately, current IDTV standards and tools are not suitable for making exhaustive use of return channels, since, at most, they provide for limited feedback from single users. For that reason, we presented in [6] a technological framework for collaborative t-learning services, extending our previous work on the development and deployment of educational applications [7]. This paper elaborates on that framework, introducing a way to define complex interaction patterns among users (Section 3) and a network infrastructure to manage the information they exchange (Section 4). Conclusions are given in Section 5.

2. A scheme for distributed t-learning services

This section outlines our framework for collaborative t-learning services (hereafter referred to as courses), describing how they are structured and the technologies we use for their design and implementation. Further details can be found in [6].

As shown in Fig 1, a course is built as a set of pedagogical units (PUs), which are aggregates of contents and programming acting as the primary level for contents organization. The PUs can be arbitrarily complex, and may contain any kind of elements (text, media clips, user interface controls, etc.), which can be laid out in multiple sceneries and scenes. The sceneries determine how the logic of a PU is distributed among the machines of users and service providers (spatial organization), whereas scenes provide for the temporal organization of activities.

Scenes were already introduced in [7], but sceneries are a specific need of distributed courses. Besides supporting distribution, they provide for the definition of roles among users, inasmuch as the elements included in the scenery run by a user determine his possible actions, and also define how those actions affect other users. For example, in a remote lecturing service, the lecturer and his audience run different sceneries, so that only the lecturer is given elements to control the sequencing of the slides.

Every scenery runs a Scenery Manager (SM), that supervises what happens in its own scenery and acts as a centralized point for communication with others. All the SMs collaborate in implementing the logic of the Course Manager (CM), a virtual entity that controls the sequencing of the PUs and supervises access to them. Furthermore, the SMs enforce the roles of the different users and control the coordinated presentation of contents.

* Work supported by the Ministerio de Educación y Ciencia, research project TSI2004-03677.
Implementation Technologies We have implemented our approach on top of the Multimedia Home Platform (MHP) standard [3], which is nowadays the reference in the normalization of IDTV applications and receiver equipment. Following the comments given in [6], our implementation is based on the DVB-J model, which we complement with several freely available technologies that fit well within it: XML, JavaBeans and JXTA [10]. Both XML and JavaBeans were introduced in the context of broadcast t-learning services in [7], and their use is slightly enhanced for distributed ones. On the contrary, JXTA is devoted to a specific need of collaborative services, as we use it to define a network architecture to support their communication needs.

We have integrated these technologies into a CASE tool that assists developers all through the design of collaborative t-learning services. This is actually an extension of the tool we presented in [7] for broadcast-based services. Among other enhancements, the new version implements the sceneries approach for distributed logic, and also supports the SCTL-MUS methodology [11] to define interaction patterns among users.

3. Defining interaction patterns with SCTL-MUS

As it was noted in [7], the programming of the courses is done by means of the event adapters mechanism of JavaBeans, linking the events that occur in certain elements of the PUs with the actions to be done in others. This approach is adequate for the type of interactions found in broadcast-based courses, including the design of user interfaces and the synchronized presentation of contents, provided that an adequate set of beans is available— together with a suitable environment for their composition. Notwithstanding, in a distributed environment there exist interactions which are difficult to model simply by interconnecting beans. First, the actions of the different users can be interleaved in an arbitrary way, being it difficult to find cause-effect relationships between the different events and the actions they must trigger (to make matters worse, many events do not always trigger the same actions). Furthermore, even though the interaction wanted for a course is conceived as a unitary whole that involves the actions of the different users, the corresponding functionality must be distributed over the different sceneries of the course; as a result, getting to the correct programming is a non-intuitive, cumbersome and error-prone task.

To solve these problems, we have resorted to specifying interaction patterns using SCTL-MUS, a formal methodology with an incremental and iterative life cycle (Figure 2) that allows getting to the desired functionality by means of successive refinements, thus reducing the complexity of a one-step design.

With SCTL-MUS, the desired interaction is defined in the “Initial goals” stage, where the developer can specify requirements about when to permit or forbid the different actions. Then, those requirements are automatically translated into a prototype, over which it is possible to automatically verify certain properties to check that the correct interaction is being defined; besides, the developer can validate the prototype by manual animation.

The developer can take as many iterations of the commented stages as needed. Then, when he finally decides that the specification is both correct and complete, the resulting prototype enters the “Implementation” stage. Here, the prototype undergoes architectural refinements, using the LOTOS process algebra [5], until allowing semi-automatic translation into code language.

SCTL-MUS admits plugins for many different languages and implementation styles, and so we have added a specific one for the structure commented in Sect. 2. This plugin is obviously based on the Java language, and works in a way...
that the functionality defined during “Initial goals” results from the collaboration of the different Scenery Managers (this is coherent with the fact that an SM supervises the actions that occur in its scenery and the messages exchanged with remote ones). This way, the role played by a user in a course is determined by i) the elements of the scenery he runs, ii) the JavaBeans interrelations among those elements and the elements in other sceneries, and iii) a piece of Java code added to the corresponding SM that conditions what the user can do at any given moment, depending on his own past actions, the past actions of other users and the events launched by the service itself.

Due to the formal approach, introducing the SCTL-MUS methodology in our tool for the development of t-learning services allows guaranteeing that the implementation of a course adheres to the desired interaction patterns, replacing the tedious manual tests for the automatic verification of properties. Furthermore, the developer only handles requirements that involve meaningful events for the service being described, so that he is freed from programming details; this is especially welcome in the IDTV field, because content and service designers are not expected to know about programming languages.

4. Supporting communication with JXTA

This section describes the network architecture we have designed to support the communication needs of a collaborative course. To this aim, it is necessary to extend the MHP communications solution, as the mechanisms offered to operate the return channel are still rather simplistic, only suitable by themselves to achieve limited feedback from individual users to service providers. Moreover, MHP assumes a rigid separation between broadcast networks and return channels, whereas it has been recently argued (see [4]) that some interesting advantages can be achieved by making transparent use of both types of networks —the idea is to combine the flexibility and bidirectionality of the Internet with the robustness and multicasting capabilities of the broadcast networks.

There are several reasons why we consider JXTA an ideal candidate to solve the commented problems. First, it fulfills the needs of the sceneries approach for distributed logic, as it can be seamlessly used on a wide range of devices (including, of course, MHP receivers). Besides, being network and language-independent, it allows establishing ad-hoc virtual networks on top of Internet and non-IP networks, thus providing for their combined use. Finally, due to its peer-to-peer basis, JXTA gives end users and devices much more relevance than in classical client-server architectures, introducing powerful mechanisms for resource discovery, group management and communication, etc.

In our approach, a JXTA virtual network is created that involves all the agents who take part in a given service (Figure 3). All the entities in that network (services, informational resources, etc.) are given an identifier, which is published by means of advertisements, i.e., XML descriptors with information about the advertised entities and pointers indicating where they can be found.

The mechanism used for discovery is based on rendezvous super-peers, i.e., peers in well-known locations designed to cache advertisements. Due to the limited computing power of set-top boxes, we suggest using service providers’ computers as rendezvous nodes (see Fig. 3). For this purpose, these nodes maintain a set of distributed hash tables with information about the users who are currently active in the different virtual communities, the roles they are playing, the resources they are accessing, etc.

JXTA defines a second type of super-peers —relays—to deal with heterogeneous networks and protocols, allowing applications to exchange messages with no concern about the networks they traverse. We exploit this idea to introduce broadcast relays, which provide for the combined use of broadcast and IP networks for communication in a service. Broadcasting is a natural option for information flows that should be served to many members of a virtual network; in fact, as noted in [9], broadcasting is currently the only possibility in MHP to transmit any audiovisual content generated by a user.

Abstracting communication issues We have designed a JXTA layer that solves the problems related to resource addressing and discovery, group communication, the selection of the medium to transmit information and the distributed management of service information.

As shown in Fig. 4, the JXTA layer varies slightly depending on the type of node. The basic implementation runs on the users’ receivers, where it allows applications to request resources only by their JXTA identifier, regardless of whether they are available in the broadcast emissions or, on the contrary, they must be accessed through the return channel. Besides, it provides a unique set of communication primitives that allow exchanging messages regardless of the medium that will be used to transmit them —messages sent through the Internet are delivered using the JXTA unicast...
and multicast pipes (see [10]), while messages to be broadcast are sent to a broadcast relay, which forwards them onto the broadcast networks addressed to the corresponding virtual network.

In rendezvous nodes, the JXTA layer includes protocols for the management of service information. Taking advantage of that information, these nodes collaborate in detecting which resources are the most utilized ones over the virtual networks. This is an interesting feature, as long as it is expectable that certain resources may be reused for services with different logics, contents and appearances (for example, the Java classes that implement scenarios for a remote lecturing service, as their functionality is quite general). The results of the surveys are periodically sent to broadcast relays, where the most popular resources are considered for inclusion in the broadcast emissions. On the receivers’ side, the JXTA layer monitors the information that is being broadcast. Thus, when an application requests a given resource, it is immediately known whether that resource is available through broadcast, in which case it can be accessed with no need to issue a query through the return channel. Clearly, this helps to speed up accesses and avoids squandering Internet bandwidth.

5. Conclusions

We have sketched a framework for the development and deployment of collaborative t-learning services, together with a selection of suitable technologies for its implementation, focusing on the definition of roles among the users and the need to support communication in an effective and efficient way.

On the one hand, we remark the adequacy of the scenarios approach to define distributed logic, and also the convenience of supplementing the visual programming promoted by JavaBeans with a formal methodology like SCTL-MUS to control the possible ways in which users can interact. SCTL-MUS allows modelling complex interaction patterns which are difficult to achieve by interconnecting beans, providing a powerful and elegant solution that allows guaranteeing correctness, whereas using only JavaBeans would make the design tasks tricky and error-prone. This kind of support is missing from the existing IDTV development environments, which also fail to support the definition of distributed logic.

On the other hand, our work comes as an extension to the mechanisms defined in the MHP standard to operate the return channel. Our proposal is based on peer-to-peer technologies, as they natively provide for the organization of users into virtual communities of people with shared interests, solving communication and group management tasks, among others. We have also discussed the convenience of establishing a virtual network on top of physical ones. In this context, the use of broadcast networks for multicast is a very interesting feature, particularly when dealing with multimedia content.

References