A multi-tiered agent-based architecture for a cooperative learning environment

E. Sánchez*, M. Lama, R. Amorim, A. Riera, J. Vila and S. Barro

Grupo de Sistemas Inteligentes (GIS)
Departamento de Electrónica y Comunicación. Facultad de Físicas
Universidad de Santiago de Compostela. Spain

Abstract: In this paper the problem of educational resource management in a cooperative learning environment is discussed. A task model was elaborated to determine both functional and level-of-service requirements. Among the former we salient: (1) the remote control of classroom hardware devices, and (2) the accessibility to course materials. Among the latter, we emphasize scalability, extensibility and reliability. A multi-tiered agent-based software architecture is proposed and a distributed deployment is presented in order to satisfy all the requirements. High-level resource management services are the key components that distinguish our architecture from others as well as prepare the proposed system for future evolution.

1. Introduction

In the last years significant efforts to introduce new technologies in classrooms have been made. On this regard, educational researchers and institutions emphasize the needs of incorporating handheld devices (also called personal digital assistants, PDAs) to common teaching and learning activities. A first use of PDAs could be to facilitate interactions in the classroom. Following this approach, there exist some applications that allow students either to answer teacher test in real time [Chen00], or to cooperate for solving a particular problem, sharing the solution at any given moment [Nanesh01]. Other uses of PDAs in educational environments regards with direct control of hardware devices as well as programs that the teacher manages in their explanations [Rekinoto00, Myers01]. Both kinds of applications are especially interesting in cooperative learning, as they favor participation and discussion between users, facilitate student activities supervision, and increase the dynamism of explanations [Hechelle00].

In this work we present a software architecture and its distributed deployment carefully design to fit on a cooperative learning environment. In this one, both teacher and students use desktop computers and/or PDA to access educational resources, such as hardware devices (projector, digital whiteboards, etc.) and course materials. The main objective of the system is to solve a resource accessibility problem: how the user can transparently access to the educational resources. As a first step, we have elaborated, by using commonKADS methodology, a tasks (or services) model that describes the resource accessibility needs of both teacher and students in order to perform their classroom activities [Lama02]. Based on this model, the next step was to determine a set of functional system requirements. Additionally, level-of-service requirements (scalability, accessibility, extensibility, reliability and security) were also carefully considered. Scalability, for instance, is a key feature in order to develop an open and flexible architecture in which new components and services can be easily added.

The paper is structured as follows: section 2 describes the educational environment as well as the system tasks model for cooperative learning; section 3 identifies system requirements; section 4 describes the software architecture; section 5 presents the distributed system design; finally, section 6 discusses the proposed architecture and compares it with other similar systems.

2. Domain and system task model

Our teaching environment is constituted by a set of hardware devices intended to support teaching and cooperative learning tasks. According to their capabilities, those devices can be classified as follows (figure 1):

- Teaching hardware resources. They are used by the teacher, and eventually by the students, in order to facilitate follow-up explanations.
  - Ninio. It acquires what is written on the blackboard and saves it as an image file.
  - Projector. It displays teacher and student presentations. The model currently used is a NEC Multisync MT-1030.

- User devices (or PDAs). They are used by human agents to manage the hardware resources of the environment, to introduce/request personal data (calendars, educational materials, etc.), or to interact with other agents in order to share information in real time (to follow-up of explanations or to look for answers to evaluation forms). As PDA we have used a Compaq iPocket.

* To whom correspondence should be addressed: ehsanchez@usc.es
• Support devices. They enable the communications amongst the user and teaching devices identified in the environment.
  - Classroom server. Its function is send/receive the information to/from Mimio and projector. It runs server-side software components that carry out low-level control of hardware resources [Riera2001].
  - Access point. It facilitates the communication among PDAs and the components of environment. We have used a Compaq WL400 model, which allows a simultaneous communication with 11 Mbps bandwidth.

Figure 1. Environment description

2.1 System task model

The system task model identifies the services that must be executed in order to perform the user operations. The model is constructed based on the teacher and student needs to access to resources in the environment. The tasks of this model can be easily classified considering the resource under their control:

• Information management. It presents the teacher the required course information: student data, list of pedagogical objectives, pre-designed activity templates, or the current educational activity. This kind of tasks also include services to solve teacher-student interactions: student requests for additional material or resources needed to perform the proposed activity; teacher requests for student answers to evaluation forms sent during the development of the educational activity or even to negotiate a common agenda with regard to the scheduling of additional lectures, and so on.

• Access to hardware resources. This means interaction with hardware resources in order to send/acquire information regarded teaching explanations. In addition, these tasks can forward information to PDAs in order to facilitate teaching follow-up.

3. Requirements

In what follows we introduce the main requirements for the project.

1. Functional requirements

   • FR-1. Manage user accounts for professors, students, groups and administrators.
   • FR-2. Provide a course-planning interface to schedule course activities and develop appropriate materials.
   • FR-3. Store course schedule, syllabus, activity templates and course materials.
   • FR-4. Allow remote control of classroom devices: projector and whiteboard acquisition system, among others.
   • FR-5. Allow access to course materials from the classroom.
   • FR-6. Allow storage of observations as well as personal notes during lectures.
   • FR-7. Perform in-classroom evaluations. The professor could be able to select a questionnaire from the course database, send it to the students, and collect student answers.
   • FR-8. Manage both professor-student and student-student communication. Every user should be able to interact with other user and/or user groups connected to the system.

2. Level-of-service requirements
4. Software architecture

In the following sections we describe the main features of the software architecture we have developed to satisfy the requirements.

4.1 Topology

The software architecture shows a multi-tiered topology in which every tier packages components offering similar capabilities. This approach aims to achieve LOS-5 requirement by means of reducing the number of different component types that each component has to interact with. Figure 2 shows a high-level view of the multi-tiered topology in which we have: (1) a Resource Tier (RT), (2) a Resource Management Service Tier (RMST), (3) an interface tier (IT), and (4) an Interface-oriented Communication Service Tier (ICST). RT packages all resource drivers available in the environment; RMST packages all Resource Management Services (RMSs), which offer both high- and low-level services to handle the resources; IT represents the set of all required graphical user interfaces; finally, ICST allows interface-interface as well as service-interface interaction.

It is important to explain why two different service tiers are considered. These are needed to encapsulate both resource management services (FR-3, FR-4, FR-5, FR-6, and FR-7 requirements) and interface-oriented communication services (FR-8 requirement), respectively. The first type can be further classified in two service subtypes: purely-resource management services (FR-3 to FR-6 requirements) and communication-involved resource management services (FR-7 requirement). The existence of different service types implies different data and control flows within the architecture (see section 4.3 for further details).

4.2 Components

Figure 3 shows the complete software architecture. Different types of components need to be introduced:

1. DB/R (Database Resource). This component represents a database manager, which handles every database operation.
2. DE/R (Device Resource). There could be different devices (from 1 to N) available in the classroom. For each one, there exist a component in charge of accessing and controlling its operation.
3. A/RMS (Administrator Resource Management Services). The system needs administration support to create accounts (FR-1 requirement), to monitor user activities, or even to check each device state. This can be done by accessing the system database.
4. P/RMS (Professor Resource Management Services). This component is a key part of our architecture. It contains all services available to the professor, not only out-of-classroom, but also in-classroom. P/RMS should facilitate pre-active and post-active teaching tasks as course planning, activity design and group design. In addition, P/RMS has to offer resource accessibility and student/group communication services.
5. S/RMS (Student Resource Management Services). It handles the services available to students. These services include device control, Internet access, and course information access among others.

6. G/RMS (Group Resource Management Services). It handles the services available to student groups. Our project requires a system ready to support collaborative learning. This is why S/RMS and its services need to be available.

1. RMS/LS (Resource Management Services Lookup Service). Before interface components can use services, they should know which services are available. Each service component is therefore associated with a lookup service. This service allows two main operations: (1) service components can register on it, and (2) interface components can discover and locate those services through it.

7. A/I (Administrator Interface). It allows the administrator to use ARM/S services (FR-1 requirement).

8. POC/I (Professor Out-of-Classroom Interface). This interface allows the professor to prepare activities, to plan the schedule, and all other activities typically performed in the pre- and post-active stage (FR-2 requirement).

9. PIC/I (Professor In-Classroom Interface). It is the interface that the professor has in the classroom during lectures. It should allow the professor to use those services typically required in the active learning stage.

10. SIC/I (Student In-Classroom Interface). It is the personal student interface.

11. GIC/I (Group In-Classroom Interface). It is the student group interface.

12. ICS/LS (Interface-oriented Communication Services Lookup Service). It is a service that allows interfaces to register in, and support interface discovery. This component is essential for professor-student, student-student and service-student interactions.

There are other components, not shown for clarity in figure 2, which also play an important role. These are repository components located in the Resource Management Service Tier. These repositories, contained on each RM/S, represent a cache memory mechanism in order to reduce the number of database accesses, thus increasing user perceived performance (LOS-2 requirement).

Figure 4 shows the internal architecture of a generic RMS. We have here a federation of agents and components that are in charge of performing services on behalf of users. Component-based services are low-level services such as switching on the projector or activating the whiteboard acquisition system. Among them we point out the validation component that is in charge of providing security support (LOS-6 requirement). Agent-based services are, on the other hand, in charge of high-level services, which were captured in the tasks model described in section 2: obtain course information, obtain current activity, obtain student evaluation, and so on. Agents were considered because each high-level service has an associated goal and a plan to achieve that goal. Furthermore, we have a service mediator agent that handles every incoming request and routes it to the appropriate service. This agent also sends back to the user the service reply.

It could be also noticed that an RMS is proposed for each type of user. The motivation behind it is to provide user scalability (LOS-4 requirement).
4.3 Connectors

We discuss here the main types of architectural connectors [Shaw96], shown in Figure 3:

1. **DE/R-S (Device Resource-Service connector).** It is an implicit invocation (also called event) connector. Services send requests to resource components and wait for corresponding notifications. The requests can be considered as events that reach the connector, which routes the event to the appropriate resource component.

2. **DB/R-S (Database Resource-Service connector).** It is a shared data (also called data access) connector. It allows access to the database manager component in order to support query requests.

3. **RMS-I (Resource Management Service-Interface connector).** It is an implicit invocation (event) connector. An interface requests certain action that is translated into events, which trigger an appropriate operation in the corresponding service.

4. **I-ICS/LS (Interface-Interface-oriented Communication Services Lookup Service connector).** Other implicit invocation (event) connector. It handles interface requests to automatically register in the ICS/LS component.

5. **RMS-ICS/LS-I (Remote Management Service-Interface-oriented Communication Services Lookup Service-Interface connector).** A common distributor connector. It is in charge of communication routing among interfaces, as well as from some RMS to the available interfaces.

4.4 Information flow

Every action in the architecture is initiated by any user selecting the desired action by means of an interface entry. Therefore, we have a user-driven mechanism. However, a more detailed discussion about information flow mechanisms is needed when the three mentioned types of services in section 4.1 are considered: purely-resource management (or type 1) services, communication-involved resource management (or type 2) services and interface-oriented communication (or type 3) services. When user requests either type 1 or type 3 services, the performed action is translated into architectural events, which, appropriately managed though implicit invocation connectors, trigger the activation of required components. Therefore, we
have an event-based mechanism underlying these operations. On the other hand, when user requests type 2 services, two information flow mechanisms are involved: (1) an event-based mechanism to handle user request, and (2) a push-based mechanism to send the obtained data to selected interface/s. Under this operation condition, the requested service acts as a broadcaster and the selected interface/s act as receiver/s.

5. Distributed System Design

Figure 5 shows the software architecture distributed in a computer network. Three key elements are linked: software components, computer hosts and network connections. We also want to point out the three main spaces involved in our project: (1) the classroom, in which the learning activity takes place; (2) the department/organization, in which organizational and database servers, as well as office desktop computers, are located, and (3) remote locations (LOS-1 requirement), such as an information service division location, to allow remote administration tasks, and professor’s home, for remote working support.

5.1 Computer support

The following computer types are considered:

1. PDA. A key device in our system. It is basic for remote device control and personal information management. We use Compaq iPaq PDAs.
2. Classroom computer. A conventional PC providing Group Interface required for collaborative learning.
3. Classroom server. A PC server running the core software components of our system. Additionally it handles access to every device in the classroom.
4. Database server. Again a PC server offering database storage and a file name system.
5. Organization server. A PC server allowing remote accessibility and security control mechanism to our system.
6. Office computer. A desktop computer providing an interface to both professors and administrators.

5.2 Network support

In the classroom we have an Ethernet LAN for classroom computer-server connections, and an access point to allow wireless connectivity from PDA. There is another Ethernet LAN in the department/organization offering connectivity among department hosts, as well as from the classroom server to the department servers. Finally,
remote accessibility is achieved through a gateway/router device, not shown in figure 4, providing Internet connection (LOS-1 requirement).

5.3 Operating system
We have a great variety of OS in our system: (1) PDA runs Windows CE, (2) classroom computers and server runs Windows 2000, (3) Database and Organization server run Red Hat Linux, (4) desktop PCs can run both Windows and Linux. At this moment, we have OS constrains only with regard to the PDAs and the classroom server.

5.4 Database
In order to minimize budget we have resorted to mySQL, a popular freeware database offering reasonable performance and recently added transaction management facilities.

5.5 Middleware
Appropriate middleware is required for component remote accessibility. We have distinguished the following middleware technologies:
- Database connectivity. JDBC is the driver chosen to develop the DB/R-S connector for remote accessibility to mySQL database.
- Remote Method Invocation (RMI). DE/R-S, RMS-I, RMS-ICS/LS-I and I-ICS/LS connectors will be developed with Java RMI. Attending LOS-2 requirement we have run RMI versus Socket connectivity tests in order to quantify the performance impact of each technology. The response time with different file sizes was minimal (milliseconds range). Therefore, we have finally chosen RMI for easier development.
- Discovery services. As it was pointed out in section 4.3, we need two discovery services: (1) for interface clients to know all available resource management services in the classroom, and (2) for every interface client and some resource management service to discover all interface clients currently connected. What we have is a spontaneous network in which clients do not know a priori: (1) possible available services, and (2) other available clients. In every discovery service a two-step operation is required: (1) services being registered in a lookup service (LS), and (2) client interfaces discovering those existing services and how to locate them. We have chosen JINI, the Java discovery service technology, as the middleware to implement the discovery services. What is more, JINI offers additional features, such as service redundancy, that have a great impact on reliability, thus providing support for LOS-3 requirement.

5.6 Programming language
Java satisfies all our needs for all components unless some DE/R components. Its networking and middleware capabilities are a commodity that will have a great impact on reducing development time. For some DE/R components, for instance the
whiteboard acquisition system, either proprietary or open-source C++ code is the only option available to handle specific hardware devices. In such cases, convenient adaptors/wrappers for these components will be developed.

5.7 Agent communication

KQML is the protocol chosen for agent communication protocol. We will use a reduced set of KQML syntax because agent-to-agent interactions are quite simple. With this approach we provide an standard communication protocol, which will facilitate future integration of new agents in our RMS federation.

5.8 Data and message modelling

XML is the language we have chosen for data and message description. XML is a standard data format and it will allow other components to talk easily with our system.

6. Discussion

The proposed architecture satisfies both functional and level-of-service requirements introduced in section 3. For each type of user there is a RMS offering a complete set of high-level resource management services. These services are the key design feature that distinguishes our architecture from others. The Pebbles project, for instance, provide access from PDAs to programs running in a PC [Myers01]. In this way, Pebbles grants PDA clients to perform low-level resource operations. Our approach is clearly more ambitious as we provide specific service tiers, which (1) provide high-level services, (2) reduce client computations, and (3) solves scalability, extensibility and reliability issues.

The continuous evolution of educational hardware resources could suggest the integration of either new services in the system or, even, new clients (such as an intelligent decision-support system for the teacher). Both situations could be solved in many different ways by adding: (1) new agents/components in existing RMSs; (2) new RMSs for a new type of users; (3) new components in the Interface Tier, and/or (4) new entries in the Interface-oriented Communication Services Tier.

Acknowledgments

Authors wish to acknowledge support from the European Commission and Spanish CICYT through grant TIC2000-0873.

References


