Blackboard Architecture to Integrate Components and Agents in Heterogeneous Distributed eLearning Systems: An Application for Learning to Program

Abstract

To build complete and complex eLearning systems, eLearning engineers are used to applying standards that facilitate sharing information as well as distributed service oriented architectures that provide reuse and interoperability by means of component integration. These concepts lead us to a Component-based Development Process that will allow us to implement tools that give full support to the teaching/learning process, taking advantage of the synergy effect created by the integration of the different components. Thus, throughout this article we analyse the proposals from the most relevant consortia concerned with eLearning standards, showing their service oriented approaches and the middleware technologies which can be used to implement them. This analysis will demonstrate that the use of middleware technologies that use the definition of services' interface can limit the reuse and interoperability requisites desired by the main standards consortia. Then, we will show a proposal which tries to solve this shortfall, using a blackboard-based architecture for integrating and communicating heterogeneous distributed components, as well as a user environment that also allows us to perform component integration. . As an example, we will demonstrate how we have built an application for learning to programme, by applying our approach and following a Component-based Development Process to implement different components (services, agents, clients, etc.) that integrate it. Hence, we will argue that using blackboard architecture and a Component-based Development Process helps us to solve the identified shortcomings.

Keywords

eLearning Standards, Eclipse, Tuple-Spaces, Component-Based Development

1 Introduction

Nowadays, the growth of Web-Based Education (WBE) environments means that groups such as IEEE LTSC (http://ltsc.ieee.org), IMS Global Learning Consortium (http://www.imsglobal.org) or ADL (http://www.adlnet.org) work to provide a set of standards allowing reusability and interoperability in the eLearning industry, setting the standard base where engineers and developers must work to achieve eLearning systems integration. That is, the current eLearning environments seek to promote reuse and allow interoperability not only among systems, but also among components. Therefore, we will be able to build tools that support the whole teaching/learning process by means of a synergistic effect achieved as a result of the integration of different components and services [IMS-AF, 2003] [IMS-SOA, 2009] [IMS-WS, 2005] [IMS-LTI, 2010].

To support the construction of these eLearning environments, nowadays we are witnessing a constant evolution of standards, specifications, reference models and best practices, which make extensive use of software engineering, model analysis, etc., in order to facilitate the implementation of systems that use them. In general, the literature refers to all these terms as "eLearning standards".

However, the implementation of standardised services and reference architectures proposed by the standardisation committees and working groups are not tied to a specific middleware platform or technology. Nevertheless, selecting the most appropriate is a fundamental task to ensure component integration and system scalability. Moreover, allowing the integration not only of services, but also of intelligent agents and components that support the teaching/learning process, and doing it from different heterogeneous processing sources, are additional objectives that we have set as an ambitious challenge in our research. It will allow us to fit together the different pieces that

make up a complete and complex eLearning system using a heterogeneous distributed architecture so that it can support the teaching/learning process during its different stages.

Developing such systems with intelligent pieces, we must point out the Intelligent Tutoring Systems (ITS) [Wenger, 1987] [Murray, 1999] [Murray 2003] and other related systems such as Adaptive Intelligent Educational Systems (AIES) and Adaptive Web Based eLearning Systems (AWBELS). ITSs are usually used together with Adaptive Hypermedia Systems (AHS) for providing "intelligent" navigation through the educative material and learning activities. Those systems that merge ITS and AHS are the so-called Adaptive Intelligent Educational Systems (AIES) which are called AWBELS when they are web-oriented. These kinds of systems allow adapting the learning process to each particular student. To do so, such systems try to analyse the students' interactions and, if possible, their solutions to assignments. In that way, they try to determine the student cognitive stage so that the learning process can be adapted to each concrete student.

In this sense, the idea of a components-based architecture allowing the reuse of existing components in ITS related systems was pointed out several years ago [McCalla, 1990] [Brusilovsky, 1995][Ritter, 1995]. Thus, we can find several proposals based on intelligent components and agents integration like those provided by McCalla [McCalla, 1990] or Ritter [Ritter 1996] for standalone computer applications, or those proposed by Ritter and Brusilovsky [Brusilovsky, 1997] [Ritter, 1998] [Brusilovsky, 2004] for distributed Web-based environments. These works manage the main idea of integrating several Web-based systems to get *"more than the sum of their parts"* [Brusilovsky, 1997]. However, in most cases, they were only theoretical approaches because the technology at that moment limited such work, as Brusilovsky points out [Brusilovsky, 2004].

In addition, the integration of components must not only be done from an architectural point of view but also from the point of view of the elements that constitute the Graphical User Interface (GUI) since the system can respond to the user interactions and give the feedback in the most suitable way [McCalla, 1990].

More recently, some authors have dealt with the application of eLearning standards and reference architectures in order to integrate eLearning systems. Thus, Dagger *et al.* [Dagger, 2007] point out the use of service oriented frameworks in order to support Learning Management Systems (LMS) composed by integrating interoperable services for the next generation of eLearning platforms. In the same way, we can find the research work carried out by Muñoz-Merino *et al.* In [Muñoz-Merino, 2009], some deficiencies in state of the art eLearning standards are identified and new services specifications are presented in order to allow integration among LMS. In [Muñoz-Merino, 2010], an architecture is described for Personal Learning Environments (PLE). PLEs allow users to build their own learning paths by using available services from different communities on the Internet [Severance, 2008]. Muñoz-Merino *et al.* recognise the integration of ITS and PLE as a desirable feature to provide personal user adaptation in the learning process. They consider service integration by means of Web Services, and propose the use of a central learning design for teachers, students and communities that "orchestrates" the learning path by using IMS-LD [IMS-LD, 2003] to specify them [Muñoz-Merino, 2010]. However, no intelligent agents are incorporated into the architecture.

Nowadays, within the framework of the aforementioned eLearning standards, the distributed middleware and component technologies, it is possible to retrieve the ideas presented by these authors, creating intelligent eLearning environments by means of components' and agents' integration. Thus, we will be able to develop complex systems that take advantage of a synergy effect. To develop those systems from software components, we can make use of Component-Based Software Engineering (CBSE) [Crnkovic, 2006] due to the possibility of managing the entire lifecycle of Component-Based products.

To focus our work and test our approaches, we have taken as our research framework the Programming Algorithms competence [ACM/IEEE, 2008] [ACM/IEEE, 2010]. In this research framework, we can find a lot of ITS approaches to learn to program, i.e. ELM-ART [Brusilovsky, 1996], Interbook [Brusilovsky, 1998], KBS-Hyperbook [Nejdl, 1999], AHA! [Bra, 2003], Problets [Kumar, 2004], ELP [Truong, 2005], Kenya Eclipse [Chatley, 2005], ProPAT [de Barros, 2005],

SICAS-W [Marcelino, 2006], SICODE [Pérez, 2006], J-LATTE [Holland, 2009], etc. Most of them are distributed Webbased or Integrated Development Environment-based applications, however none of them follows a standard-based approach nor can they profit from the synergic effect created by the integration of services and agents.

Thus, the experimental environment we have developed to test our approach has been applied to supporting the teaching/learning process for this competence by providing an environment for the users and the corresponding services and agents to provide the adaptation while students acquire this competence.

Thus, throughout this article we will explain the standardised eLearning service oriented architectures and the different middleware that can be used to implement them (Section 2). Then, as a system on which to validate the feasibility of our proposal, we will see how, within the framework of our research, we have implemented an approach based on a blackboard architecture for the integration and communication of heterogeneous distributed components, and how it has been chosen to implement different components, such as *plug-ins* for the Eclipse platform to make them part of a larger system (Section 3). After that, some implementation-related issues will be outlined (Section 4) exposing the Component-Based Development Process, requirement analysis, system architecture, etc., as well as our developed prototype system. Following this, the identified results will be pointed out (Section 5). Finally some concluding remarks will be made (Section 6).

2 Background

Several research groups work to provide a set of specifications in order to provide the principles of reuse, interoperability, accessibility and durability in the eLearning industry. Standards from LTSC IEEE, specifications described by IMS Global Consortium or reference models designed by ADL, provide the standards framework with which developers and eLearning engineers should work to achieve interoperability among systems.

Within this framework, in the following subsections we will outline some issues related to the reference architectures for eLearning systems specified by the most relevant standardisation committees as well as the middleware technologies that can be used to implement them.

2.1 Standard Reference Architectures to Build eLearning Environments

As mentioned above, from a structural point of view, there is great interest in providing standardisation to allow interoperability and reuse in eLearning environments [IMS-AF, 2003] [IMS-SOA, 2009] [IMS-WS, 2005] [IMS-LTI, 2010]. Most of the standardised specifications are outlined as services that can interact and interrelate with each other. In fact, there are reference architectures for developing services oriented eLearning systems. They are the so-called Services Oriented Architectures (SOAs) such as the IEEE Learning Technology Systems Architecture (LTSA), ADL (Advanced Distributed Learning Network) Shareable Content Object Reference Model (SCORM) [SCORM, 2004], the OKI (Open Knowledge Initiative) Framework, the JISC (Joint Information System Committee) eLearning Framework and the IMS Abstract Framework (IMS-AF) [IMS-AF, 2003].

IEEE, ADL, OKI and JISC work together with IMSs. IMSs provide specification to IEEE LTSC, the working group in charge of deciding on the adoption of a specification as standard. The ADL SCORM reference model is based on several IMS specifications. Furthermore, ADL and an integral part of OKI work as members within IMSs. JISC is an IMS member through the CETIS (Centre for eLearning Technology and Information Systems). These associations make their proposed architectures closely related, and show the same layer division and functionalities.

As an IMS Abstract Framework implementation example, we can mention the service-oriented architecture proposed by JISC. JISC, far from creating new specifications and standards, seeks to analyse, use and reach a consensus using the specifications created by other working groups. Perhaps, this makes its architecture the most pragmatic of all. In it, we can see three layers that contain a set of services:

- *The User Agents Layer*, in which we can find Learning Management System (LMS) front ends, eLearning portals, authoring tools and user applications, etc. This can be equivalent to the IMS-AF Application Layer.

- *The Learning Domain Services Layer*, which contains services like course management, learning flow, ePortfolio, etc. It is equivalent to the IMS-AF Services Application Layer.

- *The Common Services Layer*, where we can find services shared with other domains, such as authentication services, communication through chats and forums, message exchange, etc. It is equivalent to the layer of the same name inside the IMS-AF.

All these reference architectures are conceived to develop generic LMS. However, what we want to do is to allow the integration not only of generic services, but also intelligent agents and components to improve the teaching/learning process in order to build adaptive and intelligent educational systems, based on shared educational components.

As an example of eLearning architecture based on distributed reusable intelligent components, we can find the KnowledgeTree architecture pointed out by Brusilovsky [Brusilovsky, 2004]. Even though this architecture has been conceived for Web-based environments, it draws some interesting conclusions that we can incorporate into our proposal. Thus, in KnowledgeTree, Brusilovsky states that the way to eLearning classrooms for Adaptive Web-Based eLearning Systems is through a component-based architecture for adaptive eLearning. To do so, he proposes replacing the current monolithic LMS with a community of distributed communicating services (called servers by the author). Among those services, the minimum architecture must implement at least *the activity service*, *the value-adding services*, *the learning portals* and *the student model service*. In this architecture it is not only services that can be integrated and reused, but also some intelligent components that the author calls *activity services*. These services can host interactive and adaptive learning content, as well as learning services such as discussion forums, shared annotations, etc. As interactive and adaptive learning content, each activity service encloses the necessary logic and intelligent processing. In this architecture, the reuse principle cannot be performed among these intelligent components; however it gives a good starting point.

Nevertheless, Brusilovsky designed the KnowledgeTree architecture taking into account the corresponding standards to allow reusing different components by means of standardised architectures and communication protocols, so that we can place each service from the KnowledgeTree inside its corresponding JISC reference architecture layer:

- *The learning portal* provides a centralised single-login point where students and teacher work using all the learning tools. Brusilovsky envisaged this service as a Web-oriented environment. However, nothing forbids us from implementing it with any another kind of technology. Moreover, the architecture allows multiple portals but all of them can provide access to the same universe of distributed content and services. This is tied with the User Agents Layer.

- *The activity services* can host interactive and adaptive learning content, as well as learning services such as discussion forums, shared annotations, etc. These services are part of the Learning Domain Services.

- *The value-adding services* consider functionalities such as adaptive sequencing, annotations, visualisation, etc. These services are linked to the JISC Common Services Layer.

- *The student model service* is a component that represents the students' needs and the prospects in the process of eLearning in order to personalise the learning material for each individual student. This service can be part of the Learning Domain Services in the JISC reference architecture.

Table 1 shows the layers pointed out by IMS-AF and the JISC implemented architecture. In the third column, we can see the KnowledgeTree and how it fits in with IMS-AF and JISC architecture. Furthermore, the table shows some services identified for each layer.

IMS-AF	JISC	KnowledgeTree	Services
Application Layer	User Agents Layer	Learning portal	Components for the user interface, Learning Management System (LMS) front ends, eLearning portals, authoring tools and user applications, etc.
Services Application Layer	Learning Domain Services Layer	Activity services, student model service	Course management, learning activity flow, ePortfolio, student model, etc.
Common Services Layer	Common Services Layer	Value-adding services	Authentication services, communication services (chats, forums, message exchange, etc.)
Infrastructure Layer			Underlying communication infrastructure services

Table 1. Comparing some reference architectures

Thus, with this example, we can see how eLearning standards can be used to build intelligent learning environments based on service and components integration. Indeed, due to their service-oriented nature, the standardised reference architectures previously mentioned usually propose an implementation based on Web Services (WS) for communication among layers [IMS-AF, 2003] [IMS-SOA, 2009] [IMS-WS, 2005]. However, no standardisation group forces us to use any concrete middleware. In fact, we can use a lot of options to implement a system following a service-oriented architecture.

There are several proposals for service-oriented middleware that allow the construction of heterogeneous distributed environments like Web Services, CORBA, ICE, Java RMI, .Net, etc., some of which will be analysed in the next subsection. Typically, these technologies are based on providing an environment in which we can register the services and launch the objects that implement them. The middleware is responsible for providing a reference to the object or objects that implement the functionality associated with a particular service. However, to gain access to them it is necessary to know their public interfaces. In this sense, it is difficult to add services that generate added value if they are not previously defined and standardised. Some middleware technologies will be analysed in the next subsection.

2.2 Middleware Technologies for Implementing Heterogeneous Distributed Architectures

As we have pointed out previously, because of their service-oriented nature, the reference architectures tend to propose a Web Services based implementation for communicating among services within different layers (WS) (http://www.Webservices.org/). WS allows encapsulating the business logic of the components and is independent from the programming language and the platform. They work on top of HTTP protocol and use XML to encode the service requests, specified following the W3C's Simple Object Access Protocol (SOAP) recommendation. However, a WS is not a platform for services implementation, but for the service specification. WS also seem to introduce overhead compared to any benchmarked platform [Baker, 2002] [Demarey, 2005] [Gray, 2005] mainly due to the use of XML to encode the requests and the necessity of HTTP servers to allow the communication. However, using optimised XML parsers and encoding XML via binary formats should obtain better results, but this will still require improving performance greatly compared to other service-oriented middleware [Demarey, 2005].

This functionality offered by WS could also be obtained using CORBA (Common Object Request Broker Architecture) (http://www.corba.org/). In this aspect, it is necessary to mention the

effort of the project CORBAlearn [Anido, 2001a] [Anido, 2001b], of the Ibero-American Telematic Network RICOTEL, where a CORBA domain is intended to provide standardisation in the interface definition of the eLearning services.

Among CORBA's advantages, it builds a compact binary transport protocol directly on top of TCP/IP, which has a direct impact on the performance [Demarey, 2005]. Although CORBA can resolve WS deficiencies, the facts point to WS as the most supported alternative for most of the industry. In this sense, Baker [Baker, 2002] proposes a consensus solution that leads to middleware integration, so a universal solution doesn't exist but there are solutions for each problem and with the creation of bridges between middleware technologies all the solutions can be integrated and interoperable.

On the other hand, Henning from ZeroC (http://www.zeroc.com/ice.html) tries to give an explanation for this support to WS and the marginalisation of CORBA [Henning, 2006a], analysing the social, economic and technological factors that have made CORBA a technology relegated to the niche of embedded and real time systems. So, Henning states, *"CORBA was a victim of the industry's trends and fashion"* [Henning, 2006a]. Likewise, Henning points out CORBA's technical complexity, its lack of certain characteristics such as security and support for the version control, the difficulty to build a good event distribution service, the lack of asynchronous method invocation/dispatching among client-servers, and the lack of mapping to languages like C# and Visual Basic, putting CORBA outside the .NET architecture.

To solve this situation, ZeroC has developed the Internet Communication Engine (ICE). Its authors tried *"to build a middleware platform that is as powerful as CORBA, without making all of CORBA's mistakes"* [Henning, 2006b]. With it, ICE is an object oriented middleware for heterogeneous distributed environments that provides more efficient implementation in broadband, CPU and memory than WS, avoiding the CORBA complexity [Henning, 2010]. Moreover, ICE provides a set of characteristics such as security, event distribution and support for asynchronous method invocation/dispatching, among others.

Besides, we can find other middleware technologies which are more oriented towards particular platforms and programming languages. Examples of these include Java RMI (Remote Method Invocation) and the Microsoft .NET platform. The first one uses Java to achieve cross-platform services. However, it is a middleware in which the unique programming language allows specifying and deploying the services is Java. On the other hand, the .NET platform allows building heterogeneous distributed systems using different programming languages like C# or Visual Basic .Net, but both the programming language and the communication protocols must be those supported by the platform.

Feature	**Web Services**	**CORBA**	**ICE**	**Java RMI**	**.Net**
Services interface specification	Yes (WSDL)	Yes (IDL)	Yes (Slice)	Yes (Java)	Yes (.Net related languages)
Platform independent	Yes	Yes	Yes	Yes (Through JavaVM)	No
Multi-language support	Yes	Yes	Yes	No	Yes (but only for .Net related languages)

Table 2. Comparison of some middleware technologies

Table 2 summarises the features for the cited middleware technologies. So far, all the mentioned middleware technologies are based on providing a well-known and standardised definition of the services' interface. The counterpart is found in other kinds of middleware

technologies that allow the incorporation of intelligent agents and services using blackboard architecture. These are known as tuple spaces based middleware [Gelernter, 1985], in which a shared associative memory is used as a means of communication between different agents, services and clients. So, as an alternative to remote method invocations imposed (and restricted) by a public interface, the processes will write and read the information from a shared memory accessible through the network.

Furthermore, in more general terms, to choose one middleware or another for the implementation of a heterogeneous distributed service-oriented system, will depend on the characteristics required to fulfil it in terms of programming languages, issues of performance and bandwidth, architectural decisions such as topology and scalability, services that must be provided by the middleware for the deployment and maintenance of applications, platforms to be used by the different objects that compose the system, etc.

2.3 Remarks on Reference Architectures and Middleware Technologies

Whenever we build a complex system, we must seek a solution that takes into account the performance, eliminates the complexity of development and management, and that is not dependent on any platform or programming language. However, from the point of view of eLearning systems, it is also interesting to allow not only services integration with well-known and standardised public interfaces, but also intelligent agents and other clients to support the teaching/learning process. This is where the middleware based on tuple spaces comes into play. Its bases will be explained in the next section.

3 Proposal: Apply Blackboard Architecture and Component Integration

Most of the middleware mentioned so far, takes as a starting point a standard interface definition of the services that will compose an eLearning system. This principle facilitates the implementation of environments from different educational and computational paradigms (Virtual Learning, Blended Learning, Mobile Learning, etc.).

However, regarding our work, we sought a middleware to build heterogeneous distributed eLearning systems to allow us to implement them by means of the integration of the different pieces which will compose the whole system. Furthermore, such integration should not only to be restricted to those well-known public interface services previously defined and standardised, but should also allow incorporating intelligent agents and other services to support teachers and students in the teaching/learning process.

It is in this sense that architectures and middleware technology that support both the integration of services and the addition of *ad-hoc* agents should be sought. Typically, the architectures that use agents are based on the use of a communication central bus. The different components communicate by sending and receiving all the information through it. The agents "monitor" all the information that goes through the communication bus. Thus, if the information is useful for them then they collect it, manipulate it and generate the corresponding output information that is sent through the same bus.

An architecture that can be used to implement a system with these characteristics is known as blackboard architecture. This architecture is based on the blackboard metaphor. The main idea is to build and add new information in a collaborative way, starting from the information that is available on a shared basis that everyone involved can see.

The blackboard metaphor looks like this: a group of people are placed in front of a blackboard where the definition of a problem is written. Then, someone starts writing their contribution to it. In continuation, whoever who has something to add will take a turn and add to the information on the blackboard. Then someone else will take a turn and read the information generated by the previous person so that they can make their contribution to the blackboard. This step will be repeated until the solution to the problem is reached. That is, the blackboard is the common knowledge base and each specialist, starting from the problem specification and finishing by reaching the solution to the problem, iteratively updates it.

Looking for the system implementation, this metaphor seeks to benefit from the synergy created by the integration of different agents to become part of it. Small pieces with clear defined functions (agents) use the information available on the blackboard to generate new information that can be used by other agents with specific functionality.

This metaphor, which is the basis of the blackboard architecture, is used in tuple spaces based systems. These systems were introduced with the Linda coordination language [Gelernter, 1985]. A tuple space is an associative shared memory, where information is organised as a set of tuples. A tuple is the key element of the system. It consists of a vector of fields containing data type and value.

In such systems, producer agents send data to the tuple space (they write on the blackboard) and consumer agents receive data from the space if it matches with some pattern or template (read from the blackboard only the information that they understand or are interested in).

The use of tuple spaces middleware technologies has several advantages as Krummenacher *et al.* point out [Krummenacher, 2005], namely: there is no need to address communications partners directly; there is no need for synchronous links between communication partners; there is no need to run in the same computational environment as long as access to the same space is guaranteed. Furthermore, Krummenacher *et al.* assert that *"The decoupling has obvious design advantages for defining reusable, distributed, heterogeneous and agile communication applications"* [Krummenacher, 2005].

Regarding the software that allows us to store the tuple spaces, we can find the systems known as tuple space servers, which can be accessed through the network. There are different implementations of these tuple spaces servers and we can mention some of the most popular:

- TupleSpaces, developed by David Gelernter as part of Linda in the 1980's in Yale University [Gelernter, 1985]. There are implementations in Prolog, Ruby (Rinda), Python, C, Smalltalk, Java and Lisp.

- JavaSpaces (http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/), developed by Sun Microsystems and used as objects repository in Jini. It uses Java RMI (Remote Method Invocation) for the communication subsystem.

- TSpaces (http://www.almaden.ibm.com/cs/TSpaces/), developed by IBM. It is implemented in Java which is the only programming language to use the API. It has database connectivity and event notifications.

- SQLSpaces (http://sqlspaces.collide.info/), developed in the Duisburg-Essen (Germany) University by the COLLIDE (COLaborative Learning in Intelligent Distributed Environments) research group [Guiemza, 2007]. It is implemented in Java and provides a multi-language programming interface, with native APIs for several programming languages such as Java, Prolog, Ruby, PHP and C#. Furthermore, it has Web Services support so it allows the implementation of clients in other programming languages.

Feature	TupleSpaces	JavaSpaces	TSpaces	SQLSpaces
Developed by	Yale University	Sun	IBM	Duisburg-Essen University
Platform independent	Yes	No (Needs JavaVM)	No (Needs JavaVM)	Yes
Multi-language support	Yes (Prolog, Ruby, Python, C, Smalltalk, Java y	No	No	Yes (Java, Prolog, Ruby, PHP and C#)

Lisp)

Table 3. Comparison of tuple spaces servers

Table 3 summarises the features outlined for the cited tuple spaces servers. As we can see, the table considers the platform dependent feature so that we can build heterogeneous components. JavaSpaces and TSpaces are considered platform dependent because they need JavaVM, which can limit the solution. Thus, observing this table it seems that SQLSpaces is the best choice.

All these servers are based on providing a simple set of operations that allows the client to handle all the information stored on them in the form of tuples. These operations allow the agents to *write*, *read* and *take* the tuples from the associative shared memory. The primitives that implement these operations tend to be quite similar in the different implementations. Thus, generally speaking, the main function/method calls are:

- *write(tuple)*: adds or writes a tuple to a specific tuple space.
- *take(tuple_template)*: retrieves from the tuple space a tuple that matches the tuple template. The matched tuple is removed from the tuple space and returned to the invoker.
- *takeAll(tuple_template)*: has the same effect as the previous one, but this time returns all the tuples that match the tuple template.
- *waitToTake(template_tuple)*: searches for a tuple that matches the specific tuple template. If the tuple is not found in the tuple space, it blocks the execution thread until a tuple that matches the desired template is stored in the tuple space. At that time, it is removed from the tuple space and returned to the invoker.
- *read(template_tuple)*: has the same effect as *take* but the tuple is not removed from the tuple space.
- *readAll(template_tuple)*: has the same effect as *read* but returns all the tuples that match the tuple template.
- *waitToRead(template_tuple)*: has the same effect as *waitToTake* but the tuple is not removed from the tuple space.

Likewise, an interesting feature in the communication among the different agents that interact through the tuple space is event notification. With this, the server sends notification to those clients interested in receiving a specific event when a tuple that matches with a specific template has been written, updated or taken. That is, the server is not only a tuples repository, but it is also an active part of the system which allows building event driven environments.

From the ITS point of view, this kind of architecture was used by McCalla *et al.* to build SCENT-3 [McCalla, 1990], an intelligent system for learning LISP. They advocate the use of a blackboard-based architecture [McCalla, 1990] arguing: *"The blackboard data structure promotes relatively easy and rapid prototyping of the system components. Direct interfaces between components disappear since each component places its output on the blackboard and derives its input from the blackboard. Such architecture permits separating out control from the definition of components, thus allowing each entity to have a declarative task description".*

Even though the SCENT-3 approach was implemented for standalone computers, the argumentation about the use of blackboard architectures can be extrapolated to the current network-based blackboard middlewares:

a) Easy and rapid prototyping of components because the data structures (tuples) are easy to manage and to understand. Furthermore, they operate with only a few primitives (*read*, *write* and *take*).

b) Components' interfaces are not necessary due to the information exchanges performed by reading, writing and taking the information stored on the blackboard. Likewise, with the notification mechanism an event-driven model can be implemented. So, no remote method invocation is necessary.

c) Separation of control and component's definition. The components' functionality can be specified and implemented in isolation from the system's handling.

In addition, the tuple spaces have been used in several implementations of distributed collaborative eLearning systems such as Group Scribbles [Brecht, 2006] and AMENITIES [Garrido, 2006]. Thus, Group Scribbles [Brecth, 2006] is a collaborative environment that tries to provide computational support to common tools in the classroom by using IBM TSpaces as communication middleware. So, the traditional pen-based annotations, classroom blackboards, marks, etc., are substituted by their electronic equivalents, allowing traditional classrooms to add characteristics such as sharing and manipulating information in a collaborative way. On the other hand, AMENITIES (A MEthodology for aNalysis and desIgn of cooperaTIve systEmS) [Garrido, 2006] is a methodology for analysing, designing and developing collaborative environments. In addition to the necessary models to provide the analysis and design tasks, this methodology proposes a tuple space based architecture using JavaSpaces for its implementation.

In this way, the blackboard architecture based on tuple spaces seems to be appropriate for implementing eLearning systems in which an *ad-hoc* integration of different components (services and agents) is necessary. This allows the system to expand, integrate and scale, so that we can implement the "Learning Domain Services Layer" and the "Common Services Layer" mentioned in section 2.

Table 4 compares the selected tuple spaces server (SQLSpace) together with the middleware technologies analysed in section 2.2. It shows that SQLSpaces allow the components' interfaces to disappear. Furthermore, it is platform independent and provides multi-language support.

Feature	Web Services	CORBA	ICE	Java RMI	.Net	SQL Spaces
Services interface specification	Yes (WSDL)	Yes (IDL)	Yes (Slice)	Yes (Java)	Yes (.Net related languages)	***No***
Platform independent	Yes	Yes	Yes	No	No	***Yes***
Multi-language support	Yes	Yes	Yes	No	Yes (but only for .Net related languages)	***Yes***

Table 4. Comparison of some middleware technologies including SQLSpaces

However, the expansion, integration and scalability characteristics should not to be restricted to the architecture that integrates the different distributed components. It is also necessary for the elements that constitute the environment where the user interacts with the system. So, choosing a platform that allows the extension of the user interface by means of components' integration in a standardised way will allow providing the necessary mechanisms to extend and improve the tool.

That platform must be chosen taking into account the specifications for the application that the user will manage. Thus, for instance, suppose that we want to develop Web-based applications. In this context, we can choose several platforms that allow components integration by means of modules, *plug-ins* or bundles like Jboss (http://www.jboss.org), designed to give support and distribution to EJB (Enterprise Java Beans) (http://java.sun.com/products/ejb/), or OpenACS (http://www.openacs.org) based on Web Services. That is, we must find a platform that allows us to integrate components while we implement the User Agents Layer mentioned in section 2.1.

In the next subsection, we are going to show how we have applied our proposal to implement an eLearning system to acquire competence in Programming Algorithms.

4 Applying a Component-Based Development Process to Test Our Proposal

To test and validate our proposals and to illustrate how we can apply them, we have developed an eLearning system. As mentioned in the introduction, the research framework we have chosen to focus our work on is the Programming Algorithms competence. Thus, the experimental system we have developed to test our approach has been applied to support the teaching/learning process for this competence. To do so, the system provides an environment for the users and the corresponding services and agents to provide the adaptation while students acquire this competence. Throughout this section we will explain the various steps of the development process.

4.1 The Development Process

To build our system we have followed the Component-based Development (CBD) lifecycle proposed by Crnkovic. In [Crnkovic, 2006] we can see a development process that consists of an adaptation of the V development process for CBD. The activities present in the lifecycle process model are [Crnkovic, 2006]:

- Requirements analysis and specification: establishing the necessary goals and services.
- System and software design: designing the software systems architecture.
- Implementation and unit testing: building each component of the system and testing it.
- System integration: joining the different implemented and tested components.
- System verification and validation: checking the correctness of the complete system.
- Operation support and maintenance: performing the required system improvements.

As we can see, this development process takes into account the different granularity degree for the components that constitute the system as well as the testing and verification issues during the development and integration of those components. Furthermore, Crnkovic introduces the study of existing developed components before implementing them, analysing their suitability and adjustment to the project, as well as their integration capabilities. In the next subsections, we are going to outline the main issues we have taken into account while developing our testing system.

4.2 Requirements Analysis and Specification

To build the system that supports the acquisition of the Programming Algorithms competence, our aim is to provide an approximation that allows the creation of an adaptive system because, as we have explained previously, it will need the implementation of services and agents.

Thus, the main requirement we set out for the system was to provide learning activities sequencing and to adapt this sequencing to each particular student, bearing in mind the score that the student achieves on some strategic learning activities (programming assignments). The basic ITS architecture was proposed by Wenger [Wenger, 1987], who identified three necessary models, namely:

- *Domain model:* constitutes the knowledge to transmit to the student.
- *Pedagogical model:* represents the pedagogical strategies, the learning activities sequencing, etc.; that is, everything concerning the instructional design or learning flow that will lead the teaching/learning process.

- *Student model:* represents all the student related aspects that have a direct effect on the learning process, i.e. preferences, skills, knowledge, etc.

To adapt these classical models to the necessities of our system, we defined three models, namely: *the student cognitive model*, *the instructional model* and *the artefact model* [Jurado, 2007c].

On the top left-hand corner of Figure 1, we can see the *student cognitive model*. It consists of a set of evaluations for each task that the student has to solve and represents the cognitive stage for the student at every moment. This model can be matched with the *user model* proposed by Wenger [Wenger, 1987]. On the top right-hand corner, Figure 1 shows the *instructional model*. This allows specifying the instructional strategy or learning design (learning activities sequencing) to be applied. In other words, the instructional model represents the learning activity flow. It will be adapted depending on the evaluations stored in the student cognitive model. It can be matched with Wenger's *pedagogical model*. Furthermore, we do not want to limit the learning process leaving the student to work alone. As a result, we added a new requisite: the system must allow the students to perform collaborative activities, so that they can benefit from the advantage of Computer Supported Collaborative Learning (CSCL).

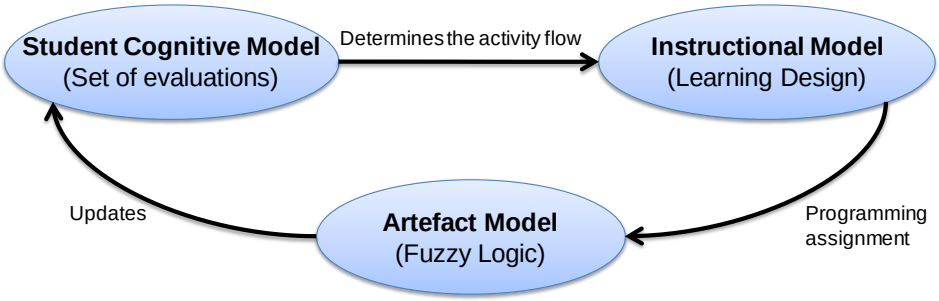

Figure 1. Models in Our Approach

Among the learning activities, a problem to be solved with an algorithm can appear. These algorithms should be analysed, assessed and evaluated. Then, to support this, a model that manages the solution must be considered. This will be the *artefact model* shown in the bottom right-hand corner of Figure 1, which is similar to Wenger's *domain model*. This model allows supporting the processing and analysing of artefacts (algorithms) developed as a solution for a proposed problem. This artefact model, which analyses the solution, interacts with the student cognitive model for updating it, reflecting the evidence of knowledge that has given shape to the solution developed by the student. In this way, in accordance with the student's work, the instructional adaptation can be achieved, deciding the next learning activity to be proposed. Thus, the learning activities are shown as a consequence of how the student solves problems.

From the user's point of view, the system must provide an environment that is ideal for developing the programming assignments. However, the best environment to perform a concrete learning activity is not always the most suitable to allow all the desired learning scenarios. For instance, for programming assignments an integrated development environment (IDE) seems to be the most appropriate. However, if the students must perform a reading activity or watch a video with the lecture and they do not have the IDE available, they may prefer an environment that allows them to view contents wherever they are, perhaps on a Web browser. So we set, as another requisite, the creation of several working environments allowing the users to choose the most appropriate in each learning scenario.

Having identified the requirements for the system, in the next subsection we will outline the architecture we designed as well as pointing out the necessary services and components to fit the requirements.

4.3 System Architecture

To define the system architecture, we have followed the recommendations proposed by the

standardisation working groups. So, we designed a multilayer architecture like the one which the JISC puts forward, with a "User Agents Layer", a "Learning Domain Layer" and a "Common Service Layer". Thus, the architecture of our system is the one exposed in Figure 2.

In the figure, we can see the two kinds of users or roles that will use the system, namely teacher and student. They can interact with the system and use the services by means of two kinds of user environments that constitute the Users Agent Layer, specifically an Eclipse-based environment and a Web-based environment. The Eclipse-based environment will offer advanced programming capabilities. On the other hand, the Web-based environment will provide an alternative and easy-to-access user interface for the same services.

To support the teaching/learning process, we have identified a set of services that are categorised in two layers: Learning Domain Services and Common Services. In the first group of services we have implemented two alternatives for sequencing the learning activities, namely CopperCore and Tuple-LD. With these two alternative services we try to analyse how easy it is to interchange services with the same functionalities without taking into account the service interface.

As stated in the requirements, among the learning activities the system proposes solving a programming assignment. So, we need a service (or agent) that provides the corresponding evaluation that updates the student model so that the learning activities sequencing can be adapted.

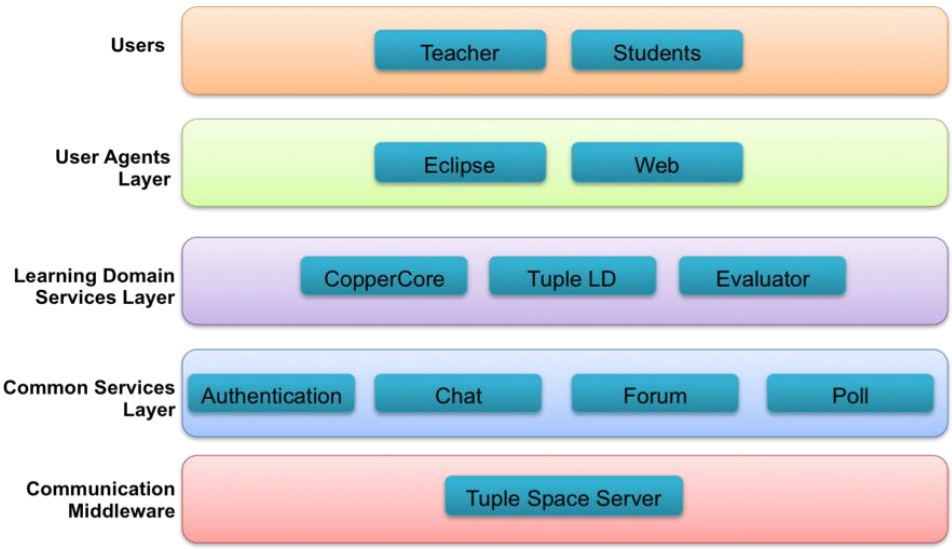

Figure 2. System Architecture

So far, the explained services (and agents) constitute the Learning Domain Service Layer. In addition, other services to provide authentication, and to support CSCL like chat, forum and polls have been implemented. They are categorised as Common Services.

The communication middleware can be seen at the bottom of Figure 2. As mentioned in section 3, the one we have chosen is a tuple space server, so that the different entities (agents and services) will interact together by means of tuples. Furthermore, this middleware will provide the storage needs that the system may have. Among the information stored in the server we can find user session data, communication messages used in different tools (chat, forum, etc.), learning activities related information, learning activities sequencing related information, students' scores on assignments, etc.

Having outlined the system architecture and identified the necessary pieces, we can have a look at some parts of it. Thus, we will go into detail about each service and agent, explaining their specific features and implementations issues.

4.4 Components' Implementation and Integration

In this section, we are going to explain each component that constitutes the previously stated architecture and to expose the main decisions for each one, as well as to show how the selected

middleware ties together all the pieces in the integration phase of the development process.

4.4.1 The User Agents Layer: Two Different User Environments

Because we have chosen as our research framework the Algorithms Programming competence [ACM/IEEE, 2008] [ACM/IEEE, 2010], in order to support the acquisition of this competence, our aim is to implement an environment that fits the following requirements:

a) To facilitate the use of different views;

b) To provide mechanisms to allow us to extend and improve the tool;

c) To use a development environment as close as possible to what the students will find in a business environment.

All these characteristics are available in Eclipse (http://www.eclipse.org). This is a widely used Integrated Development Environment [BZ Research, 2008] [Eclipse, 2009]. It is an open source project that follows the components-based recommendations proposed by OSGi (Open Services Gateway Initiative), the *OSGi Services Platform*. This platform is based on the division of the applications into little modules or bundles that can be remotely launched, stopped, installed, updated and removed without restarting. The platform is built on top of the Java Virtual Machine. The platform builds a set of layers that can be used by the corresponding bundles over the virtual machine. Bundles are little applications with a simple well-defined functionality. This functionality can be extended or used by other bundles. Eclipse adopts the OSGI bundles calling them *plug-ins*. So, we can use the functionality of the Eclipse platform and implement extensions by means of an API.

Eclipse is a complete development environment for different programming languages, which include Java, C/C++, Ruby, etc. Furthermore, it can be used for modelling designs and validating models by means of the Eclipse Modelling Framework (EMF) and the Graphical Modelling Framework (GMF). With this, we have the ideal tool to implement our proposal, since in that way we are able to check it with different programming languages and aspects related with Programming and Software Engineering.

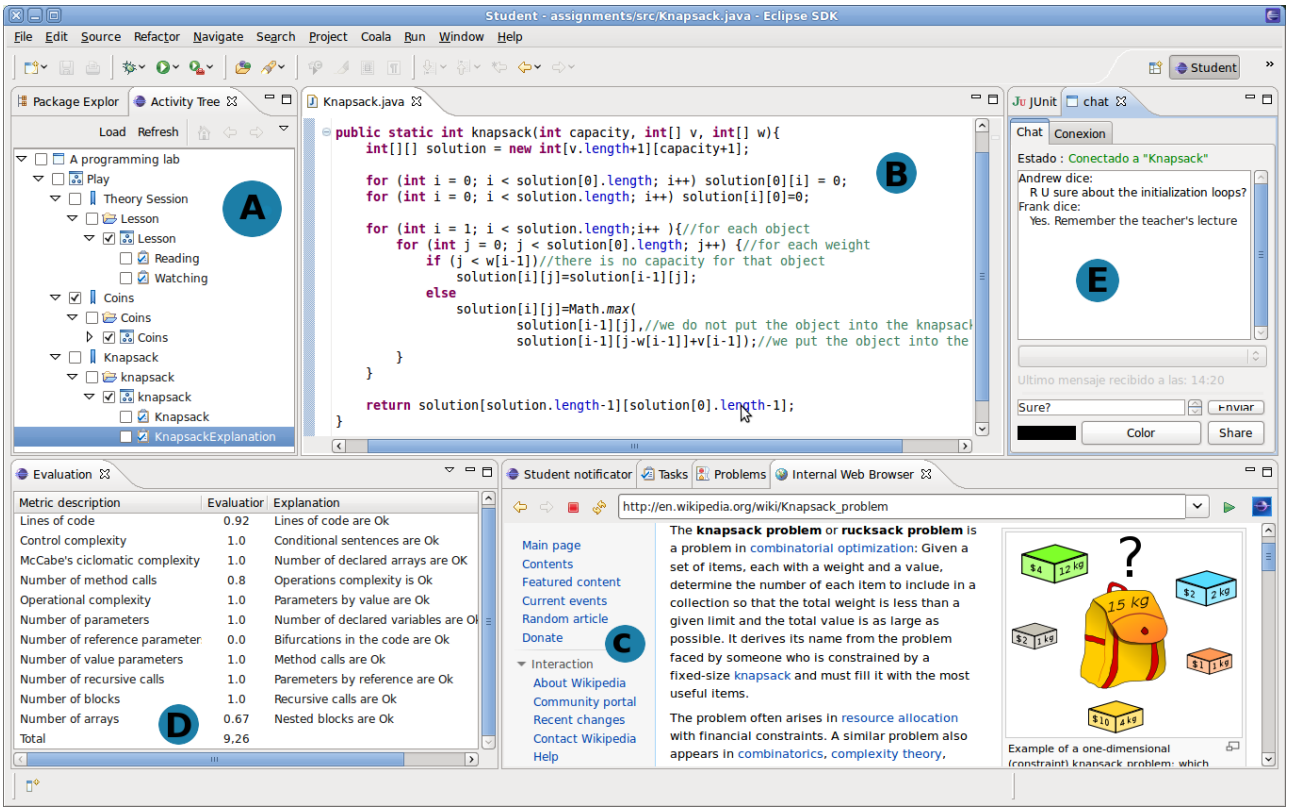

Figure 3. Eclipse-based environment with the tree activity, the assignment, the chat, the evaluation for the assignment and the support reading to perform the assignment

In these terms, we have implemented a set of *plug-ins* for the Eclipse environment. We have called this customised environment COALA (COmputer Application for Learning Algorithms) (http://chico.esi.uclm.es/coala) [Jurado, 2009]. The COALA *plug-ins* connect with the tuple space server so they can use the services and agents of the system. Figure 3 shows the appearance of the Eclipse environment once the COALA *plug-ins* are loaded. In that figure, we can see how the environment has some views and editors opened for the student perspective. Thus, the environment has the tree activities view (Figure 3 A) with the learning activities suggested by the system using the corresponding learning activities sequencing services (both CopperCore and Tuple-LD). Among those activities the student can find a programming assignment to be solved with the corresponding algorithm (Figure 3 B). Furthermore, the system can provide another learning activity (i.e. a reading activity) to support the assignment (Figure 3 C). After finishing the programming assignment, the student can ask the evaluator agent for the corresponding score and explanation (Figure 3 D). In addition, if the teacher has specified in the instructional design that the programming assignment must be done in a collaborative way, the system will open the corresponding collaborative services to use the chat, forum and polls from COLE-Programming (Figure 3 D). In the same way, the COALA teacher perspective has the necessary editors and views that allow them to set up the instructional design, the evaluation parameters and messages, watch the student progress, etc.

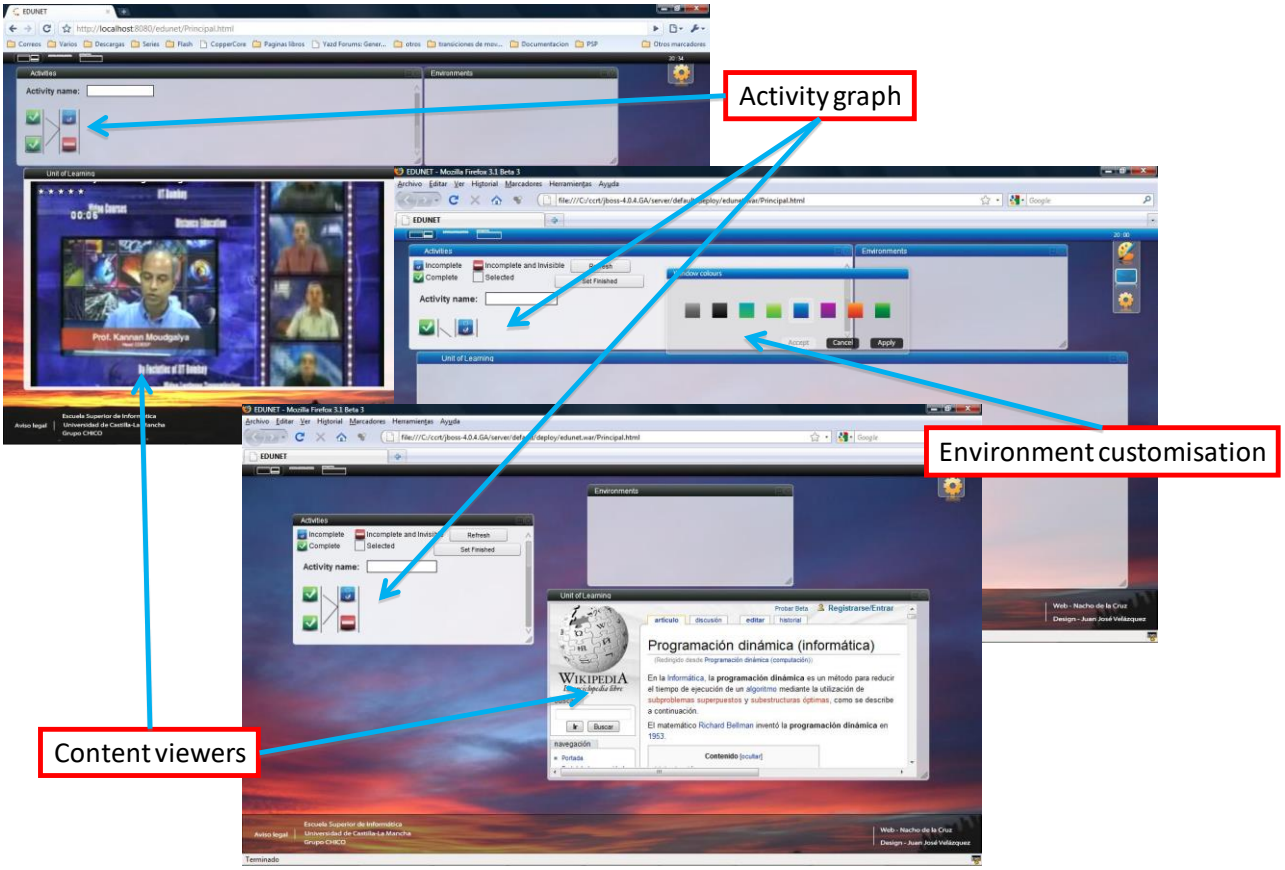

Figure 4. Web-based environment with enriched interface

On the other hand, Edunet (Education in Internet) (http://chico.esi.uclm.es/coala/index.php/Edunet) represents the Web-based environment. As COALA does, it allows running instructional designs provided by CopperCore and Tuple-LD services, as well as using the chat, forum and poll services. The main feature of Edunet is its graphic user interface designed to allow an easy user interaction, leaving the Eclipse environment. To implement Edunet we have used Adobe Flash CS4 (http://www.adobe.com/products/flash/). This has allowed us to build a highly customised environment (Figure 4). This feature makes the learning process more attractive for the students because it allows:

- Performing learning activity tasks in the environment (on the left). Those activities include content view, use of chat and forums, visualising and using learning objects. Furthermore, we have added a simple code editor so that the students can perform programming tasks.

- Viewing the activity sequencing by using the activity graph, instead of the commonly used activity tree.

- Customising the environment by modifying the window size and position, the environment skin, the background image, etc. (on the right).

The integration of Edunet into the system allows the use of the system in more learning scenarios, i.e. Blended Learning, so that users can interact with the system by using Eclipse or a Web environment that is easily accessible by a browser.

4.4.2 The Learning Domain Services Layer

4.4.2.1 The Learning-Activity Sequencing Engine: CopperCore proxy and Tuple-LD

As we have previously stated in the requirements analysis, we need learning activity

sequencing. Taking into account the Crnkovic recommendations [Crnkovic, 2006] for the system and software design in a CBD process, we analysed existing proposals that allow us to specify learning activities sequencing (so-called instructional design) in a standard way. The first alternative we have chosen is a standard-based engine.

In this sense, IMS-LD [IMS-LD, 2003] is the most widespread standard for providing the specification of instructional designs. To run this kind of specification, it is necessary to have an engine in charge to analyse and execute the instructional design for each user. The reference implementation engine for IMS-LD is CopperCore (http://www.coppercore.org). Thus, one of the services we have implemented consists of a CopperCore proxy.

This proxy transforms the notification events from the tuple space server in the corresponding method invocation to the CopperCore engine. The fields of the tuple will be processed as parameters for the method invocation. In the same way, the proxy will transform the output of the method invocation in the form of a tuple, and will write it in the server. This written tuple will generate the corresponding notification for the interested server.

On the other hand, we have implemented our customised learning activity sequencing Engine, that we have called Tuple-LD (Tuple Learning Design) (http://chico.esi.uclm.es/coala/index.php/Tuple-LD); another learning activity sequencing service. With Tuple-LD, we can specify the activity sequence following a tuple-based language that specifies Event Condition Action (ECA) rules. It includes, on the one hand, the tools and services that the teacher needs to set out the learning activities sequences and launch the runs for the students and, on the other hand, the necessary tools and services to allow the students to execute the learning designs (follow the learning activities) and allow the teacher to monitor the students' progress. Therefore, both the learning activities sequencing service and the user interface where students have a guided running of the tasks (reading, implementation of algorithms, etc.) are available

Tuple-LD uses the tuple spaces server to store and share all the information related to the specification of instructional designs, users, groups, tasks, roles, and running of the learning units. The Tuple-LD's user interface is implemented as an Eclipse *plug-in*, featuring two perspectives:

- The administration perspective: it allows adding users, groups, learning tasks, specifying instructional design and launches the running.

- The running perspective: it allows running the instructional designs inside the environment. The user interface is composed of the learning activity tree, where the learning activities are shown by using different colours indicating the state (done, to do, running, etc.); the activities that the user is doing (on the right); and some instructional design progress related statistics for the user (at the bottom).

Neither the CopperCore proxy nor Tuple-LD is restricted to a concrete user environment. As a consequence, the learning activities can be performed in every environment connected to the middleware.

During the learning activities running in the selected engine, a programming assignment can be proposed. As explained above, the adaptation process requires an artefact model that allows analysing the students' solutions and marking them. In the next subsection, we will explain briefly how our evaluator agent that implements this model works.

4.4.2.2 The Evaluator Agent

Depending on the kind of assignment we want the system to evaluate, we must implement the corresponding evaluator agent. We can implement and add all the necessary agents to evaluate the corresponding assignments.

In our system, we have chosen to evaluate the algorithms developed by the students as solutions to the programming assignments proposed by the system. To implement the agent, we have followed the proposal explained in [Jurado, 2007a] [Jurado, 2007b] and briefly shown in Figure 5. In this figure, the teacher writes an implementation for the ideal approximate algorithm that solves a

problem (on the top left of the figure). Next, several software metrics that shape its functionality will be calculated. In this way, we obtain an instance of the ideal approximated algorithm. After that, the fuzzy set for each metric will be established in the following way: initially, each fuzzy set will be a default trapezoidal function around the metric value from the approximate algorithm; the teacher can adapt each fuzzy set to indicate whether an algorithm is correct or not. From this, we obtain a collection of fuzzy sets that characterise the algorithm. Thus, we get a fuzzy representation of that ideal approximated algorithm; that is, we obtain an ideal approximated algorithm fuzzy representation that solves a concrete problem.

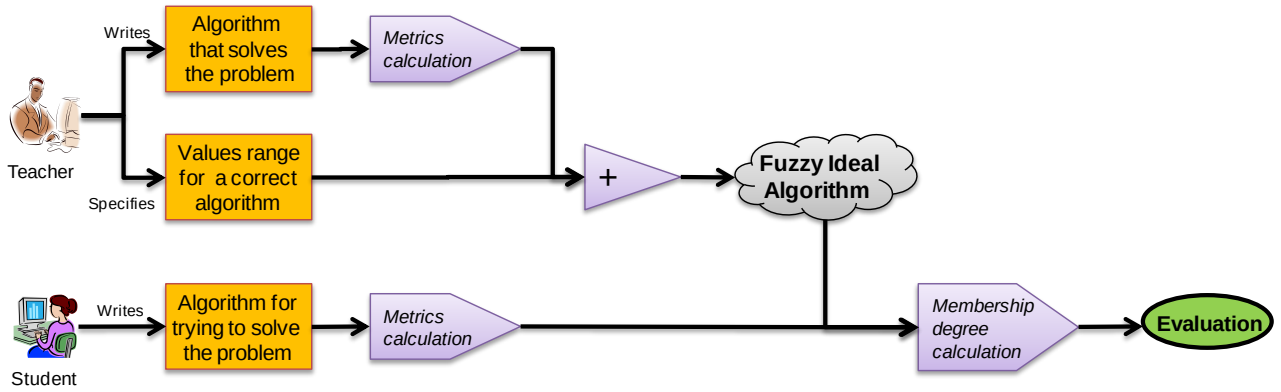

Figure 5. Process for Evaluating the Student's Algorithm

Algorithms that students have written (on the bottom left of the Figure 5) will be correct if they are instances of that ideal algorithm fuzzy representation. Knowing the degree of membership for each software metric, obtained from the algorithm written by students in the correspondent fuzzy set for the ideal approximated algorithm fuzzy representation, will give us an idea of the quality of the algorithm that the students have developed. Thus, the system will have the evaluation (on the right of the Figure 5) of the algorithm developed by the student as feedback. This evaluation consists of a score and an explanation of what is right and wrong in the student solution. This data can be read by the User Agents so that they process them in the best way.

Among the learning activities that the student can perform, some of them require collaboration with other students, integrating the necessary services and components into the user interface so that activities can be implemented in isolation and then integrated into the whole system following our suggested proposal and architecture.

4.4.3 The Common Services Layer: Some Collaborative Supported Tools

Allowing the students to perform collaborative activities so that they can benefit from the advantage of the Computer Supported Collaborative Learning (CSCL) is a feature that we must take into account. With this in mind, we developed COLE-Programming (Collaborative Learning Programming) (http://chico.esi.uclm.es/coala/index.php/COLE-programming). COLE-Programming implements a set of services and tools (chat, forum and poll) to perform information exchanges among the students who are working on a group programming assignment. The use of these tools will enrich the programming activities, allowing the use of the system in collaborative learning environments. Furthermore, COLE-Programming implements a set of features explicitly for programming, namely: a) message filtering in forums, depending on the type (problem to solve, solution proposal, discussion, etc.); b) code, warning and error messages exchange; c) collaboration statistics visualisation, like number of contributions for a specific user in each tool, participation percentages, other users with which the user has to collaborate or interchange information, etc.

COLE-Programming can be used in isolation to perform collaborative programming tasks, or joined to a learning design engine. This allows us to build an environment in which students can run learning activities sequencing and, at a certain moment a collaborative task can be proposed by the system. However, if the users do not want to use activity sequencing they can perform any

programming activity in a collaborative way. This is possible thanks to the proposed architecture. Hence, the system that integrates all the mentioned services and agents can profit from all the functionalities.

Once we have implemented and tested each component, the development process leads us to perform the system integration by joining the components. In the next subsection, we will explain how to "glue together" all the pieces of our architecture by using the proposed middleware.

4.4.4 The Communication Middleware

From an implementation point of view, among the available tuple space servers we have chosen one that allows us to use (but is not limited to) the Java language, because it will be the one used to implement the Eclipse *plug-ins*. Thus, we have chosen SQLSpaces [Giemza, 2007] because it is available under an open source license and gives multi-language support. This will provide the system with greater capacity for integration and interoperability.

As mentioned in section 3, the tuple space server will be an essential piece in the system because all the elements (user agents, services, etc.) are interconnected through it. That is, they all share the information and communicate using the write, read and take primitives of the tuple space server described in section 3.

To exemplify how to perform this communication, figure 6 shows a UML sequence diagram with the different communication messages (write, read, take and notification on tuples) among COALA's instances for teachers and students, the evaluator agent and the LD engines by means of the tuple spaces server during a typical working session. Thus, the different steps are:

- *Step 1:* As shown in the figure, in the beginning the teacher specifies an assignment using his/her COALA environment. To do so, he/she must specify the task description, some test cases the algorithm must pass, and the fuzzy ideal algorithm representation. After this, the teacher's environment uploads this assignment by sending a tuple with the form *<task id, task description, test cases, fuzzy representation>*. At that moment, the task is available to all the students in the classroom by reading this tuple from the tuple space server.

- *Step 2:* The tasks the teacher had uploaded must be those specified in one of the LD engines (Tuple-LD or CopperCore). For each student, the LD engines send a tuple with the form *<user id, run id, activity tree>* to the tuple space server which contains the corresponding activity tree. At this time, all the tasks and the corresponding activity trees for each student are available.

- *Step 3:* Once all the information is available in the tuple space server, the students' client can download their activity tree specification using the "Learning Design" view in their working environment. To do so, the client will read a tuple with the form *<user id, run id, activity tree>* that matches the corresponding user.

- *Step 4:* Following this activity tree, the students are able to download the corresponding programming assignment onto their workspace by reading the tuple *<task id, task description, test cases, fuzzy representation>* previously uploaded by the teacher, using the corresponding menu in the plug-in. Then, each student can work out the task by writing the code, compiling, etc.

- *Step 5:* Once the students have finished the assignment, they can send their solution to the server. To do so, the corresponding student's client sends a tuple with the following form *<use id; task id; solution code>*.

- *Step 6:* The tuple space server will notify the teacher's client about the solution sent by the student. Then, he/she can check the code written by the student on his/her computer by reading all the tuples with the form *<use id; task id; solution code>*.

- *Step 7:* Likethe teacher, the evaluator agent will be notified by the tuple space server about the solution the students have sent so that it can processes the code to obtain the set of metrics and an evaluation explanation. These calculated metrics are sent to the tuple space server with the form *<task id; user id; metric1; metric2; ... metricN>*. Also, an explanation associated with each metric is sent in a tuple with the following format: *<task id; user id; explain metric1; explain metric2; ... explain metricN>*.

- *Step 8:* At that moment, the tuple space server will notify the teacher, the students and the LD engine about the availability of the evaluation (software metrics and the corresponding explanations). So, teacher and students can check the automatic evaluation. At the same time, the LD engine will read the evaluation and update the activity tree for the concrete user writing the new *<user_id, run_id, activity_tree>* tuple to the tuple space. The update in the activity-tree tuple fires a notification to the student's COALA instance. This notification informs COALA that the new activity tree is available and will be downloaded. Thus, the student can follow his/her activity tree and download the next task.

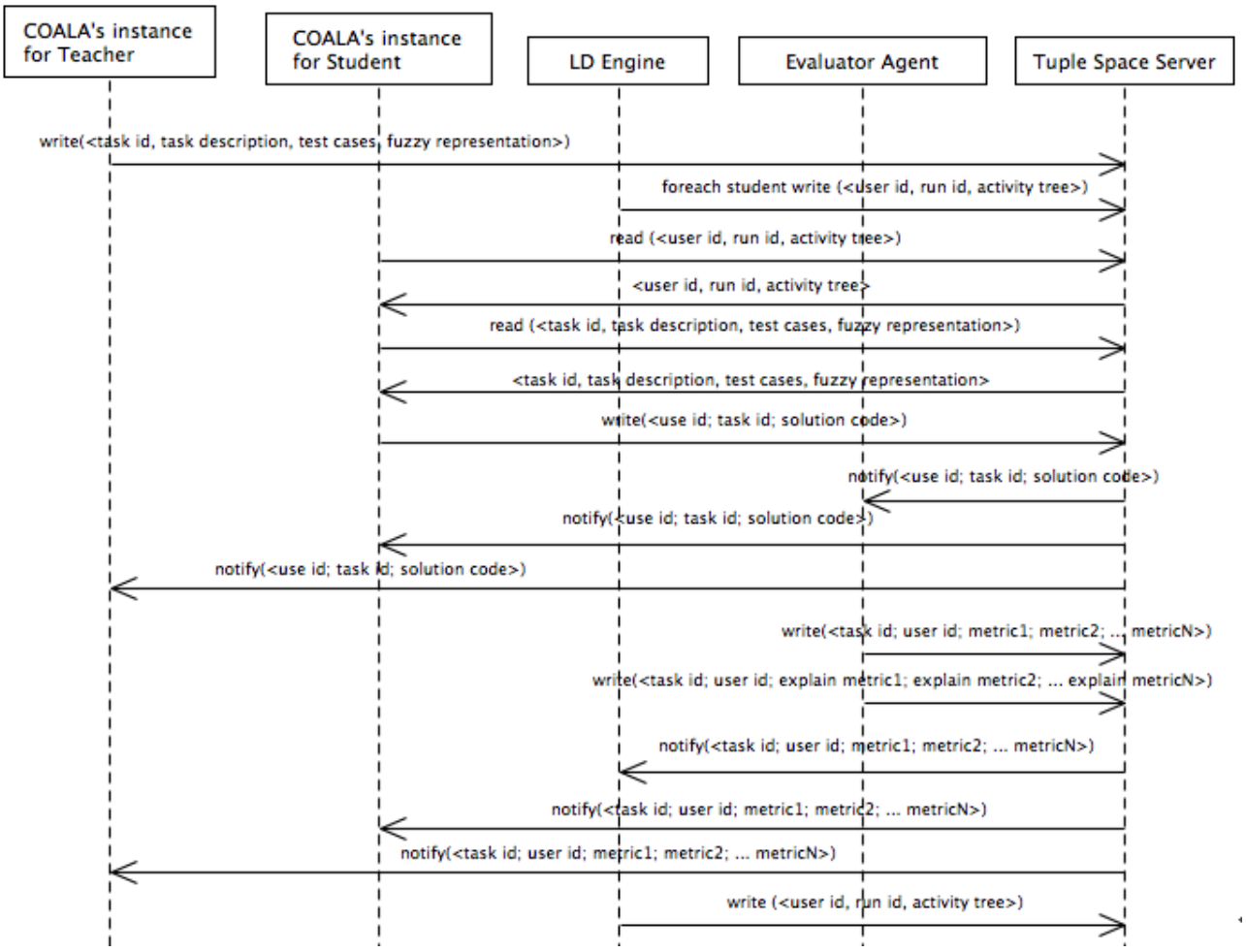

Figure 6. Tuple (message) exchange in the Blackboard Architecture

5 Summarising the Results

Throughout the previous section, the system developed to test and validate our proposals has been exposed. We have explained the CBD process, which was that followed, as well as given details about its main steps, namely requirement analysis, system architecture and component implementation and integration.

With this, we have tried to show that it is possible to build an eLearning system that follows

the eLearning standards directives using a component-based approach, both for the architecture and the user environment. To achieve that, we used a tuple space server that implements the blackboard architecture and component-based user environments.

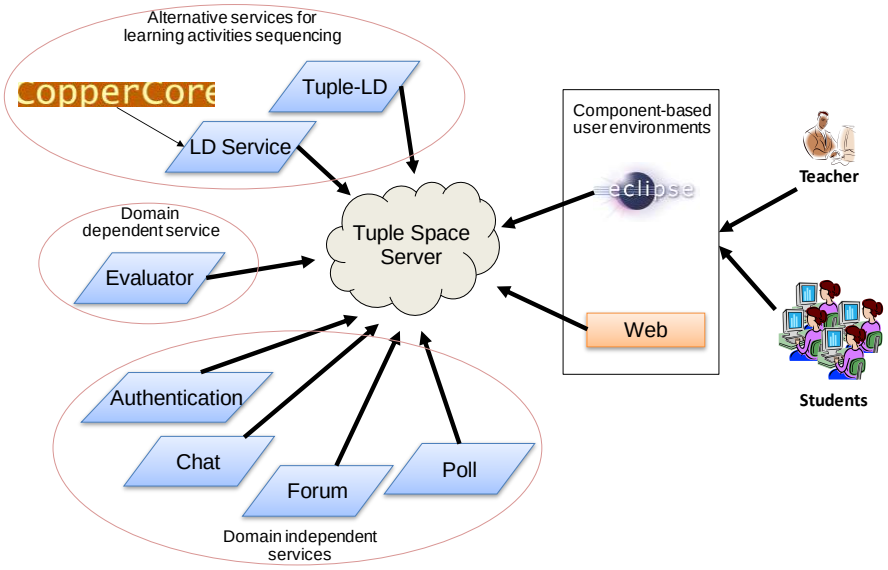

Figure 7. System Design Model

In Figure , we can see a summary of the system in which the tuple space server constitutes the central piece in the system. The server's satellites are the different agents and services it interconnects. The users can interact with the system by using two different environments, both using a component-based approach (on the right of the figure). The first one is an Eclipse based environment that follows the OSGi indications for a component-based environment. The second is a web environment. Using the tuple space server, these environments can manage the same services and agents, as the only thing they must use is the information stored in the tuple space server in the form of a tuple.

Among the services and agents the user environments can use, we have developed two alternative services for learning activities sequencing so that the system can provide the corresponding learning adaptation. These have allowed us to test the separation between control and component definition because the functionality is isolated from the service specification. In addition, this helps us to see that the service interface is not necessary, as services share information (assignments specifications, users, profiles, etc.) and they are interested in the same notifications from the tuple space server, but the functionality is implemented in different ways (using an eLearning standard for our particular implementation).

Other examples of these concepts are the domain independent services (at the bottom-left of the figure). These services can be used in other domains and by any user environment, as we have tested in our implementation.

The only domain dependent service implemented allows evaluating algorithms developed by the students. Adding or substituting this service depending on the learning domain will allow us to assess any kind of assignment, introduce better adaption techniques, implement recommending agents, etc.

So, we have shown how the integration of components can be done, both from an architectural point of view as well as from the point of view of the elements that constitute the user environment. Therefore, it is possible to build eLearning systems that follow the eLearning standards directives and support the whole teaching/learning by means of a synergistic effect achieved as a result of the integration of different components and services, using tuple spaces and component-based user environments.

6 Concluding Remarks

In order to allow building services oriented architectures that support eLearning systems following the guidelines proposed by the main eLearning standard committees, throughout this work we have introduced some of the main middleware technologies to put together heterogeneous distributed applications. With this, we can integrate services and agents on demand, allowing the construction of complex eLearning systems that can become more than the sum of their components; that is creating a synergic effect among the components.

We have exposed those service oriented middleware technologies based on the definition of a well-known public interface. Their main drawback is that it is necessary to know the communication interface (public method definition) to make use of the services, so a standardised definition is essential. Therefore, the services that compose the system must be previously known, denying the use of those services and agents that can be added on demand.

Thus, we have proposed as a solution the tuple spaces middleware technologies that use blackboard architecture. This will allow *ad-hoc* service integration by exchanging information in a centralised shared memory. So, we have shown different tuple spaces servers that facilitate the implementation labour using a simple and reduced set of primitives.

We have outlined a proposal to build eLearning systems, bearing in mind the reuse and integration of components, allowing the implementation of tools to support the whole teaching/learning process and taking profit from the integration of different components. This proposal makes use of blackboard architecture to integrate different heterogeneous distributed components by means of tuple spaces and component-based environments to build a fully extensible user interface.

Then, we have shown how we have applied a Component-based Development Process in order to build an eLearning system to acquire the competence of Programming Algorithms, so that we can test and validate our proposal. To do so, we have explained how we have implemented and integrated all the necessary services and agents to support the teaching/learning process under blackboard architecture. Then, we have shown the components added (integrated) to our user environment, thanks to the use of the proposed technologies and architecture.

Finally, we have summarised the obtained results and confirmed that it is possible to build eLearning systems that support the whole teaching/learning process by means of a synergistic effect as a result of the integration of different components and services.

As future work, a study of the system performance is highly recommended in order to contrast the latency, round-trip, scalability, etc. of the architecture and the selected middleware. Furthermore, a user-usability study may prove rather interesting, in order to test how the proposal suits the users' necessities during their learning process.

References

[ACM/IEEE, 2008] ACM/IEEE Computer Society: Computer Science Curriculum 2008: An Interim Revision of CS 2001. Report from the Interim Review Task Force. (2008)

[ACM/IEEE, 2010] ACM/IEEE: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. Report from the Interim Review Task Force. (2010)

[Anido, 2001a] Anido, L.; C., J. P. & Llamas, M.: Aplicación del Proceso Unificado de Modelado al Diseño de Arquitecturas Software para Sistemas de Teleformación Dirigidos por Estándares, in 'Primer Congreso Iberoamericano de Telemática CITA 2001'. (2001)

[Anido, 2001b] Anido, L.; Llamas, M.; Fernández, M.; Rodríguez, J.; Caeiro, M. & Santos, J.: A Standards-driven Open Architecture for Learning Systems, in 'Proceedings of the IEEE International Conference on Advanced Learning Techniques (ICALT'01)'. (2001)

[Baker, 2002] Baker, S.: Web Services and CORBA, in Meersman, R. & Tari, Z., ed., 'On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE', Springer-Verlang Berlin Heidelberg, pp. 618.632. (2002)

[Barros, 2005] Barros, L. N.; dos Santos Mota, A. P.; Delgado, K. V. & Matsumoto, P. M.: A Tool for Programming Learning with Pedagogical Patterns, in, 'Eclipse '05: Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange ', ACM, pp. 125--129. (2005)

[Bra, 2003] Bra, P. D.; Aerts, A.; Berden, B.; de Lange, B.; Rousseau, B.; Santic, T.; Smits, D. & Stash, N.: AHA! The Adaptive Hypermedia Architecture, in, 'Proceeding of HT03'. (2003)

[Brecht, 2006] Brecht, J.; DiGiano, C.; Patton, C.; Tatar, D.; Chaudhury, S. R.and Roschelle, J. & Davis, K.: Coordinating Networked Learning Activities with a General Purpose Interface, in 'Proceedings of the 5th World Conference on Mobile Learning'. (2006)

[Brusilovsky, 1995] Brusilovsky, P.: Intelligent Learning Environments for Programming: The Case for Integration and Adaptation. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education, Washington, DC, AACE pp. 1-8. (1995). Available online at http://www.contrib.andrew.cmu.edu/~plb/papers/AIED-95.html

[Brusilovsky, 1997] Brusilovsky, P., Ritter, S., Swchwarz, E.: Distributed Intelligent Tutoring on the Web. Proceedings of the 8th World Conferrence of the AIED Society, Kobe, Japan, pp. 18-22. (1997). Available onlye at http://www.contrib.andrew.cmu.ecu/…

[Brusilovsky, 1996] Brusilovsky, P.; Schwarz, E. W. & Weber, G.: ELM-ART: An Intelligent Tutoring System on World Wide Web, in, 'ITS '96: Proceedings of the Third International Conference on Intelligent Tutoring Systems' Vol. 1086, Springer-Verlag, pp. 261-269. (1996)

[Brusilovsky, 1998] Brusilovsky, P.; Eklund, J. & Schwarz, E.: Web-based Education for All: A Tool for Development Adaptive Courseware, in Computer Networks and ISDN Systems Vol. 30, pp. 291--300. (1998)

[Brusilovsky, 2004] Brusilovsky, P.: KnowledgeTree: A Distributed Architecture for Adaptive eLearning, 'WWW 2004: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters' , ACM Press , pp. 104-113 . (2004)

[BZ Research, 2008] BZ Research, 5th Annual Eclipse Adoption Study November 2008. (2008)

[Chatley, 2005] Chatley, R. & Timbul, T.: KenyaEclipse: Learning to Program in Eclipse, in SIGSOFT Softw. Eng. Notes Vol. 30, ACM, pp. 245-248. (2005)

[Crnkovic, 2006] Crnkovic, I.; Larsson, S. & Chaudron, M.: Component-Based Development process and component lifecycle, in 'Prodeedings of the ICSEA'06', IEEE. (2006)

[Eclipse, 2009] Eclipse Fundation, The Open Source Developer Report, May 2009. Eclipse Community Survey. (2009)

[Dagger, 2007] Dagger, D.; O'Connor, A.; Lawless, S.; Walsh, E. & Wade, V.P.: Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services, in IEEE Internet Computing Vol. 11, pp. 28–35. (2007)

[Demarey, 2005] Demarey, C.; Harbonnier, G.; Rouvoy, R. & Merle, P.: Benchmarking the Round-Trip Latency of Various Java-Based Middleware Platforms, in 'SIU2005', pp. 7-17. (2005)

[Garrido, 2006] Garrido, J.; Noguera, M.; Gonzalez, M.; Gea, M. & Hurtado, M.: Leveraging the Linda Coordination Model for a Groupware Architecture Implementation, in 'Proceedings of 12th International Workshop on Groupware' Vol. 4154, Springer LNCS, pp. 286-301. (2006)

[Gray, 2005] Gray, N.: Performance of Java Middleware - Java RMI, JAXRPC, and CORBA, in, 'The Sixth Australasian Workshop on Software and System Architectures (AWSA 2005)'. (2005)

[Gelernter, 1985] Gelernter, D.: 'Generative Communication in Linda', ACM Transactions on Programming Languages and Systems 7 (1), pp. 80-112. (1985)

[Giemza, 2007] Giemza, A.; Weinbrenner, S. & Engler, J.and Hoppe, H.: Tuple Spaces as Flexible Integration Platform for Distributed Learning Environments, in 'Proceedings of ICCE 2007', pp. 313-320. (2007)

[Henning, 2006a] Henning, M.: The Rise and Fall of CORBA, in ACM Queue. (2006)

[Henning, 2006b] Henning, M. & Spruiell, M.: Distributed Programming with Ice, Technical Report, ZeroC, Inc. (2006)

[Henning, 2010] Henning, M. & Spruiell, M.: Choosing Middleware: Why Performance and Scalability Do (And Do Not) Matter, in. (2010)

[Holland, 2009] Holland, J.; Mitrovic, A. & Martin, B.: J-LATTE: a Constraint-based Tutor for Java, in Kong, S.; Ogata, H.; Arnseth, H.; Chan, C.; Hirashima, T.; Klett, F.; Lee, J.; Liu, C.; Looi, C.; Milrad, M.; Nakabayashi K. Mitrovic, A.; Wong, S. & Yang, S., ed., 'Proceedings of the 17th International Conference on Computers in Education', pp. 142-146. (2009)

[IMS-AF, 2003] IMS Abstract Framework: White Paper, Technical Report, IMS Global Learning Consortium Inc. Available online at http://www.imsglobal.org/af/afv1p0/imsafwhitepaperv1p0.html. (Last visited in June 2010). (2003)

[IMS-SOA, 2009] IMS Service Oriented Architecture (SOA): Adoption of Service Oriented Architecture for Enterprise Systems in Education: Recommended Practices, Technical Report, IMS Global Learning Consortium Inc. Available online at http://www.imsglobal.org/soa/soawpv1p0/imsSOAWhitePaper_v1p0.html (Last visited in June 2010). (2009)

[IMS-WS, 2005] IMS General Web Services Base Profile, Technical Report, IMS Global Learning Consortium Inc. Available online at http://www.imsglobal.org/gws/index.html. (Last visited in June 2010). (2005)

[IMS-LD, 2003] IMS Learning Design. Information Model, Best Practice and Implementation Guide, XML Binding, Schemas. Version 1.0 Final Specification. Technical Report, IMS Global Learning Consortium Inc. Available online at http://www.imsglobal.org/learningdesign/index.html. (Last visited in July 2010). (2003)

[IMS-LTI, 2010] IMS Learning Tools Interoperability Basic LTI Implementation Guide. Version 1.0 Final', Technical report, IMS Global Learning Consortium Inc. Available online at http://www.imsglobal.org/lti/blti/bltiv1p0/ltiBLTIimgv1p0.html. (Last visited in July 2010). (2010)

[Jurado, 2007a] Jurado, F.; Redondo, M. A. & Ortega, M., “Fuzzy Algorithm Representation for its Application in Intelligent Tutoring Systems for the Learning of Programming”, in Rogério PC do Nascimento; Amine Gerqia; Patricio Serendero & Eduardo Carrillo (ed.), EuroAmerican Conference On Telematics and Information Systems, EATIS'07 ACM-DL Proceeding, Association for Computing Machinery, Inc (ACM), Faro, Portugal, 2007

[Jurado, 2007b] Jurado, F.; Redondo, M. A. & Ortega, M., “Applying Approximate Reasoning Techniques for the Assessment of Algorithms in Intelligent Tutoring Systems for Learning Programming” (in Spanish), in Isabel Fernandez de Castro (ed.), VII Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación (Sintice'07), Thomson, Zaragoza, Spain, 2007, pp. 145-153

[Jurado, 2007c] Jurado, F.; Redondo, M. A. & Ortega, M., “An Architecture to Support Programming Algorithm Learning by Problem Solving”, in Emilio Corchado; Juan M.Corchado & Ajith Abraham, (ed.), Innovations in Hybrid Intelligent Systems, Proceedings of Hybrid Artificial Intelligent Systems (HAIS07), Springer Berlin Heidelberg New York, Salamanca, Spain, 2007, pp. 470-477

[Jurado, 2009] Jurado, F.; Molina, A. I.; Redondo, M. A.; Ortega, M.; Giemza, A.; Bollen, L. & Hoppe, H. U.: Learning to Program with COALA, a Distributed Computer Assisted Environment, Journal of Universal Computer Science Vol. 15, Verlag der Technischen Universität Graz, Austria, in cooperation with Know-Center, Graz, Austria, and IICM , Graz University of Technology, Austria, pp. 1472-1485 (2009)

[Krummenacher, 2005] Krummenacher, R.; Hepp, M.; Polleres, A.; Bussler, C. & Fensel, D.: WWW or What Is Wrong with Web Services, in Web Services, European Conference on Vol. 0, IEEE Computer Society, pp. 235-243. (2005)

[Kumar, 2004] Kumar, A.: Using Online Tutors for Learning - What do Students Think?, in, 'Proceedings of Frontiers in Education Conference (FIE 2004)', IEEE, pp. 524-528. (2004)

[Marcelino, 2006] Marcelino, M.; Tosheva, I.; Dobrodzhaliev, Y.; Gomes, A. & Mendes, A.: A Web-based Approach to Support Initial Algorithmic Procedural Programming Learning, in, '3rd E-Learning Conference', pp. 101-106. (2006)

[McCalla, 1990] McCalla, G. I., Greer, J. E., & Scent Research Team.: SCENT-3: An Architecture for Intelligent Advising in Problem-Solving Domains. In C. Frasson & G. Gauthier (Eds.), Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education (pp. 140-161). Norwood: Ablex Publishing. (1990)

[Muñoz-Merino, 2009] Muñoz-Merino, P.J., Delgado-Kloos, C. and Fernández-Naranjo, J.: Enabling Interoperability for LMS Educational Services. Computer Standards and Interfaces, vol. 31:2, p. 484-498. (2009)

[Muñoz-Merino, 2010] Muñoz-Organero, M., Muñoz-Merino, P. J. and Delgado-Kloos, C. Personalised Service-Oriented E-Learning Environments, IEEE Internet Computing 14 (2). (2010)

[Murray, 1999] Murray, T.: 'Authoring Intelligent Tutoring Systems: An Analysis of theState of the Art', International Journal of Artificial Intelligence in Education 10, 98-129. (1999)

[Murray, 2003] Murray, T.: An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art, Kluwer Publishers, chapter 17, pp. 491-544. (2003)

[Nejdl, 1999] Nejdl, W. & Wolpers, M.: KBS Hyperbook - A Data-Driven Information System on the Web, in, '8th International World Wide Web conference (WWW8)'. (1999)

[Pérez, 2006] Pérez, J. R. P.; del Puerto Paule Ruiz, M. & Lovelle, J. M. C.: SICODE: A Collaborative Tool for Learning of Software Development, in, 'IV International Conference on Multimedia and Information & Communication Technologies in Education (m-ICTE2006)'. (2006)

[Ritter, 1995] Ritter, S. and Koedinger, K. R.: Towards Lightweight Tutoring Agents. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education, Washington, DC, AACE, pp 91-98 (1995)

[Ritter, 1996] Ritter, S. & Koedinger, K., 'An Architecture for Plug-In Tutor Agents', Journal of Artificial Intelligence in Education 7, 315-347. (1996)

[Ritter, 1998] Ritter, S.; Brusilovsky, P. & Medvedeva, O. Creating More Versatile Intelligent Learning Environments with a Component-Based Architecture, In 'Proceedings of the ITS98', Boettl, B.; Halff, H.; Redfield, C. & Shute, V., ed., Lecture Notes in Compute Science, Vol. 1452, Springer Verlag, Berling. pp. 554-563 (1998)

[SCORM, 2004] SCORM 2004. Overview. ADL (Advanced Distributed Learning). (2004)

[Severance, 2008] Severance, C.; Hardin, J. & Whyte, A.: “The Coming Functionality Mash-Up in Personal LearningEnvironments ,”, in Interactive Learning Environments Vol. 16, pp. 47–62. (2008)

[Truong, 2005] Truong, N.; Roe, P. & Bancroft, P.: Automated Feedback for “Fill in the Gap” Programming Exercises, in, 'Australasian Computing Education Conference'. (2005)

[Wenger, 1987] Wenger, E.: Artificial Intelligence and Tutoring Systems, Morgan Kaufman Publisher (1987)