CLPL: Providing software infrastructure for the systematic and effective construction of complex collaborative learning systems

Santi Caballé\textsuperscript{a,}\textsuperscript{*}, Fatos Xhafa\textsuperscript{b,1}

\textsuperscript{a} Dept. of Computer Science, Multimedia and Telecommunication, Open University of Catalonia, Rambla Poblenou, 156, 08018 Barcelona, Spain
\textsuperscript{b} Dept. of Languages and Informatics Systems, Technical University of Catalonia, Campus Nord, Ed. Omega, C/Jordi Girona 1-3, 08034 Barcelona, Spain

\textbf{A R T I C L E   I N F O}

Article history:
Received 25 November 2009
Received in revised form 3 June 2010
Accepted 3 June 2010
Available online 16 June 2010

\textbf{Keywords:}
Software architecture and design
Software engineering methods
Software reuse
Component-based software engineering
Model-driven engineering
Service orientation
SOA
Computer-supported collaborative learning
E-learning
Software and systems education

\textbf{A B S T R A C T}

Over the last decade, e-Learning and in particular Computer-Supported Collaborative Learning (CSCL) needs have been evolving accordingly with more and more demanding pedagogical and technological requirements. As a result, high customization and flexibility are a must in this context, meaning that collaborative learning practices need to be continuously adapted, adjusted, and personalized to each specific target learning group. These very demanding needs of the CSCL domain represent a great challenge for the research community on software development to satisfy.

This contribution presents and evaluates a previous research effort in the form of a generic software infrastructure called Collaborative Learning Purpose Library (CLPL) with the aim of meeting the current and demanding needs found in the CSCL domain. To this end, we experiment with the CLPL in order to offer an advanced reuse-based service-oriented software engineering methodology for developing CSCL applications in an effective and timely fashion. A validation process is provided by reporting on the use of the CLPL platform as the primary resource for the Master’s thesis courses at the Open University of Catalonia when developing complex software applications in the CSCL domain.

The ultimate aim of the whole research is to yield effective CSCL software systems capable of supporting and enhancing the current on-line collaborative learning practices.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Over the last years, e-learning and in particular CSCL needs have been evolving accordingly with more and more demanding pedagogical and technological requirements. Current educational organizations’ needs involve extending and moving to highly customized learning and teaching forms in timely fashion, each incorporating its own pedagogical approach, each targeting a specific learning goal, and each incorporating its specific resources. Moreover, organizations’ demands include a cost-effective integration of legacy and separated learning systems, from different institutions, departments and courses, which are implemented in different languages, supported by heterogeneous platforms and distributed everywhere, to name some of them (Atefeh and Lockemann, 2006).

As a result, modern CSCL environments no longer depend on homogeneous groups, static content and resources, and single pedagogies, but high customization and flexibility are a must in this context, meaning that collaborative learning practices need to be continuously adapted, adjusted, and personalized to each specific target learning group. These very demanding needs represent a great challenge for the CSCL research community to satisfy. Therefore, a generic, robust, flexible, interoperable, reusable computational model that meets the fundamental functional needs shared by most of collaborative learning experiences is largely expected by the research community and industry (Czarnecki and Eisenecker, 2000). Indeed, CSCL applications are extensively used by all forms of higher education and especially in online distance education where open universities have a central role and use CSCL tools massively in all their formation cycles.

Due to this extensive use, CSCL becomes very attractive for domain software developers who have recently provided a number of architecture solutions with the aim of reusing the large number of common requirements shared by e-learning and CSCL applications (Pahl, 2007). Common needs in CSCL include support for three essential aspects of collaboration, namely coordination, collaboration and communication; with communication being the base for reaching coordination and collaboration in synchronous (i.e., cooperation at the same time) or asynchronous (i.e., cooperation at different times) collaboration modes (Roseman and Greenberg, 1996). In addition, the representation and analysis of group learning activity interaction forms one of the paradigmatic principles of the CSCL domain (Dillenbourg, 1999a) and should form part of...
the very rationale of all CSCL applications (Martinez et al., 2003). Finally, in order to improve collaboration in a group it is essential to provide measures and rules to resolve authentication and authorization issues and to protect the system from intentional or accidental ill use as well as to perform all the system control and maintenance for the correct administration of the system.

Generic platforms, frameworks and components are normally developed for the construction of complex software systems through software reuse techniques, such as generic programming, domain-based analysis, feature modeling, service-oriented architecture, and so on (Czarnecki and Eisenecker, 2000; Bacelo Blois and Becker, 2002; Gomaa, 2005). Indeed, in the context of generic architectures and platforms, software reuse is by far one of the main concerns in the software industry and it is increasingly recognized its strategic importance in terms of productivity, quality and cost (Czarnecki, 2005).

However, despite the advance in software reuse, reuse capacity is still in an incipient status, mainly due to the short in scope of the reuse techniques such as classes, components, and frameworks, also so-called “reuse in the small”. There is, therefore, a need for increasing the level of reuse by extending the scope and, as a consequence, the impact on the software development, also so called “reuse in the large” (Ateveh and Lockemann, 2006). This is chiefly fulfilled by extracting the commonality and variability features of systems given a specific, wide domain and then reusing them for the construction of single systems in the same domain (Gomaa, 2005). Thus, neither longer is necessary to “reinvent the wheel” nor to develop a new system from scratch. This way, organizations can consolidate and adapt their existing key software assets to meet the ever changing requirements and needs. These approaches have been successfully applied to different domains thus providing cost-effective applications of increased quality in timely fashion. The rapid change and evolution of requirements in the CSCL domain raises new challenges to software developers, who in turn demands more powerful reuse-based software techniques that provide more flexible, adaptable, modular, and maintainable software.

Therefore, leveraging the latest software reuse principles, a generic service-oriented component-based computational model in the collaborative learning context is intended to form the very rationale of complex CSCL environments in a wide range of learning situations and pedagogical goals. As a result, domain developers can derive specific CSCL applications by systematically adapting and tailoring this reusable computational model for the construction of effective, affordable and timely newly CSCL tools, which are modular, flexible, interoperable and maintainable, and a fast adaptation of existing applications to newly learning and teaching requirements (Caballé and Xhafa, 2003).

Based on these principles, in a previous work (Caballé et al., 2007) we proposed an innovative approach in the form of a software infrastructure for collaborative learning with the aim of meeting the current and demanding needs found in the CSCL domain. In this current work, we evaluate this software infrastructure as an advanced reuse-based software engineering methodology for developing CSCL applications in an effective and timely fashion. The validation process of the effects of this approach is provided by the online software development courses found in the real context of the Open University of Catalonia.

The development of the resulting ideas of this research represents an attractive but quite laborious challenge that will yield CSCL systems capable of providing more effective answers on how to improve and enhance the online collaborative learning experience as well as to achieve a more effective collaboration (McGrath, 1991; Sfard, 1998; Soller, 2001; Webb, 1992).

The paper is organized as follows. Section 2 presents the aims and the theoretical background to the research and the development of our study. Section 3 describes the collection methodologies and adopted analysis procedures for elaboration on the resulting data. Section 4 analyses and discusses on the results obtained from the validation processes. The paper concludes by summarizing the main ideas of this contribution and outlining ongoing and further research.

2. Aims and background

In this section, a brief overview of the existing technologies and paradigms related to this work is presented, namely computer-supported collaborative learning, generic programming, service-oriented architecture, and model-driven architecture. This overview will serve as background for the next sections and becomes the very rationale of the CSCL software infrastructure presented in this paper.

2.1. Computer-supported collaborative learning

Computer-supported collaborative learning (CSCL) is one of the most influencing research paradigms dedicated to improve teaching and learning with the help of modern information and communication technology (Koschmann, 1996; Dillenbourg, 1999a; Strijbos et al., 2006; Stahl, 2006; Daradoumis et al., 2006). Collaborative or group learning refers to instructional methods and methodologies where students are encouraged to work together on learning tasks. As an example, project-based collaborative learning proves to be a very successful method to that end (Dillenbourg, 1999b). Therefore, CSCL applications aim to create virtual collaborative learning environments where students, teachers, tutors, etc., are able to cooperate with each other in order to accomplish a common learning goal.

To achieve this goal, CSCL applications provide support to three essential aspects of collaboration, namely coordination, collaboration and communication; with communication being the base for reaching coordination and collaboration (Roseman and Greenberg, 1996). Collaboration and communication might be synchronous or asynchronous. The former means cooperation at the same time and the shared resource will not typically have a lifespan beyond the sharing while the latter means cooperation at different times being the shared resource stored in a persistent support.

2.2. Generic programming

In all advanced forms of engineering it can be observed that new products are usually developed by reusing tried and tested parts rather than developing them from scratch. The reuse of previously created product parts leads to reduced costs and improved productivity and quality to such an extent that industrial processes will take a great leap forward. Generic programming (GP) (Czarnecki and Eisenecker, 2000) has emerged over the last years to facilitate this possibility in the software engineering field. GP is an innovative paradigm that attempts to make software as general as possible without losing efficiency. It achieves its goal by identifying interrelated high-level family from a common requirement set. By the application of this technique, especially in design phases, software is developed offering a high degree of abstraction which is applicable to a wide range of situations and domains. By applying GP to develop computer software important objectives are achieved (Caballé and Xhafa, 2003):

• Reuse. This means to be able to reuse and extend software components widely so that it adapts to a great number of interrelated problems.
• Quality. Here “quality” refers to the correctness and robustness of implementation which provides the required degree of reliability.
2.3. Service-oriented architecture

Service-oriented architecture (SOA) (W3C, 2004) represents the next step in the software development to help organizations meet their ever more complex set of needs and challenges, especially in distributed systems (Guiling et al., 2005). This is achieved by dynamically discovering and invoking the appropriate services to perform a request from heterogeneous environments, regardless of the details and differences of these environments. By making the service independent from the context, SOA provides software with important non-functional capabilities for distributed environments (such as scalability, heterogeneity and openness), and makes the integration processes much easier to achieve.

SOA relies on services. According to W3C (W3C, 2004), a service is a set of actions that form a coherent whole from the point of view of service providers and service requesters. In other words, services represent the behavior provided by a provider and used by any requesters based only on the interface contract. Within SOA, services:

- stress location transparency by allowing services to be implemented, replicated and moved to other machines without the requester’s knowledge;
- enable dynamic access as services are located, bound and invoked at runtime;
- promote interoperability making it possible for different organizations supported by heterogeneous hardware and software platforms to share and use the same services;
- facilitate integration of other existing systems and thus protect previous investments (e.g., legacy assets);
- rely on encapsulation as they are independent from other services and their context;
- enhance flexibility by allowing services to be replaced without causing repercussions on the underlying systems involved;
- foster composition from other finer-grained services.

Although SOA can be realized with other technologies, over the last few years Web-services has come to play a major role in SOA due to lower costs of integration along with flexibility and simplification of configuration. The core structure of Web-services is formed by a set of widely adopted protocols and standards, such as XML, SOAP, WSDL, and UDDI (W3C, 2004), which provide a suitable technology to implement the key requirements of SOA. This is so because these protocols allow a service to be platform- and language- independent, dynamically located and invoked, interoperable over different organization networks, and supported by large organizations (e.g., W3C consortium).

At the moment, there are two main Web-services approaches: SOAP and REST. The core structure of the SOAP is formed by a set of protocols used by most standards, such as HTTP, XML, WSDL, and UDDI (W3C, 2004), which provide a suitable technology to implement the key requirements of SOA. SOAP includes a wide range of WS-* specifications in support for building a composable architecture to form an environment for complex Web-service applications. A much simpler Web-service approach is the resource-oriented architecture (ROA) and REST (Richardson and Ruby, 2007), which are a lightweight alternative to SOA/SOAP that rely on a stateless, client-server, cacheable simple HTTP-oriented protocol.

There is a lot of discussion on SOAP vs. REST (zur Muehlen et al., 2005) with pros and cons on both sides being the conclusion on that either should be selected depending on the type of application to be developed. The general consensus is that REST should be used for simpler Web applications while SOAP is a more mature and complex solution to focus on critical and enterprise developments. Main points to reach that consensus are:

- SOAP is a widely adopted standard. REST is not though it does use standards (e.g., HTTP, URL, XML, etc.).
- SOAP’s most uses of HTTP as a transport protocol are incorrect and use other transport protocols. REST deals only with standard HTTP operations and any HTTP enabled application can consume REST Web-services.
- SOAP needs toolkits to deal with the development complexity. REST is easy to build and no development tools are required.
- SOAP/WS-* provide full security, reliability, and transactability. REST assumes the application deals with communication failures and supports just point-to-point security.
- SOAP relies on WSDL to describe services in terms of interfaces. REST is cumbersome to consume due to lack of definition, which means more dependence on written documentation.
- SOAP is cumbersome for simple Web applications. REST simplicity is at technological level only and the developer needs to write raw HTTP calls.
- SOAP is RPC-oriented and fits well advanced forms of engineering techniques based on models and domains. REST is resource-oriented and does neither follow OOP ideas nor fit well popular software engineering methodologies.
- SOAP’s learning curve is higher than REST but it is better investment because of adhering standards the development is faster and of more quality.
- SOAP may face low performance. REST gets benefited from caching though in more heavily loaded networks only.

2.4. Model-driven architecture

The model-driven development (MDD) paradigm and the framework supporting it, namely model-driven architecture (MDA) (OMG, 2006) have been recently attracting a lot of attention given that it allows software developers and organizations to capture every important aspect of a software system through appropriate models (Gomma, 2004). MDA provides great advantages in terms of complete support to the whole cycle development, cost reduction, software quality, reusability, independence from the technology, integration with existing systems, scalability and robustness, flexible evolution of software and standardization, as it is supported by the object management group (OMG, 2006).

In proposing MDA, two key ideas have had significant influence in OMG aiming at addressing the current challenges in software development (Karam et al., 2008): SOA and PLA. As to the former, SOA provides great flexibility to system architectures by organiz-
Fig. 1. The essential aspects in any collaborative learning (CSCL) application.
2.6. Reusable software infrastructure for CSCL applications

Generic platforms, frameworks and components are normally developed for the construction of complex software systems through the reuse technique. This approach has been successfully applied to different domains thus providing applications of increased quality, and reduced cost and development time. However, main representative research efforts following a generic approach come as an extension of the CSCW domain, and only a few of them have been extended for educational purposes (see Caballé et al., 2007; Caballé, 2008b for an extensive overview of related work). Therefore, the application of reuse-based software engineering methodologies to the specific development of CSCL applications has been, to the best of our knowledge, little investigated.

A revision of the latest research to provide generic software support to the development of applications within the CSCL domain shows that the results are still scarce. Main focus is still on leveraging the great research efforts and technological advances within the CSCW domain (Penichet et al., 2010; Fonseca et al., 2009; Petropoulakis and Flood, 2007; Bao-Qing et al., 2007; Lukosch and Schümmer, 2006). These approaches provide exhaustive support for cooperative work, such as group and workflow management, group editing, document sharing and many types of both synchronous and asynchronous communication (Fonseca et al., 2009). However, many of them are not even prepared to support essential collaborative learning features, such as collaborative knowledge building and scaffolding as well as specific monitoring and assessment of the learning process (Dillenbourg, 1999b). Representative researchers (Stahl, 2006) argue whether intrinsic CSCL requirements should be considered from the very beginning of the development and not as an extension to experimented cooperative tools for work (Bentley et al., 1997).

To sum up, from our view, current approaches fail in providing appropriate response to the two main objectives of our research: (i) leverage advanced software engineering methodologies to provide generic CSCL platforms that focus on the development process and yield effective and quality CSCL applications while saving development time and effort, and (ii) clear distinction between CSCL and CSCW domain and thus provide specific solutions to respond the intrinsic requirements of the CSCL domain from the very beginning of the development.

Therefore, the main contribution of our whole research (Caballé et al., 2007; Caballé, 2008b) is a generic, reusable, robust, flexible, interoperable, component-based and service-oriented platform called CLPL.2 The CLPL is based on the GP paradigm so as to enable a complete and effective reutilization of its generic components as the skeleton for the construction of potentially most of CSCL applications. This generic platform implements the conceptualization of the fundamental needs existing in representative collaborative learning experiences.

In order to meet these requirements, the development of the CLPL is based on the MDD paradigm and the framework supporting it, namely MDA (Czarnecki, 2005). In proposing MDA, the CLPL development takes advantage of two key ideas that have had significant influence in addressing the current challenges in software development (Caballé, 2008a): SOA and PLM.

In particular, in developing the CLPL, a PIM was first created by applying the following GP ideas (see Caballé and Xhafa, 2003): (i) define the semantics of the properties and domain concepts, (ii) extract and specify the common and variable properties and their dependencies in the form of abstractions found in the CSCL domain, and (iii) isolate the fundamental parts in the form of abstractions from which the basic requirements were obtained, analyzed and designed as a traditional three-layer architecture (i.e., presentation, business and information).

In order to achieve these goals, first, the PIM was expressed using UML as the standard modeling language promoted by the OMG (see Fig. 2). Second, two different PSM have been constructed so far from the PIM: a Java implementation in the form of a generic component-based library and a service-oriented approach by using Web-services technology.

The architecture of the CLPL (Caballé et al., 2007) is made up of five components (see Fig. 3) handling user management, administration, security, knowledge management, and functionality, which map the essential issues involved in potentially most of collaborative learning applications (see Caballé, 2008b for a detailed description of each component):

- **CSCL user management component**: this contains all the behavior related to user management in applications, which can act as a group coordinator, group member, group-entity and system administrator. It will tackle both the basic user management functions in a learning environment (namely registration, deregistration, modifications, joining a group, or meeting group members) and the user profile management. The latter implements the user and group models within a collaborative environment.
- **CSCL security management component**: this contains all the generic descriptions of the measures and rules decided upon to resolve authentication and authorization issues and so protect the system from both unknown users and the intentional or accidental ill use of its resources. It also contains the means to manage the user session and the state representation of the system.
- **CSCL administration management component**: this contains the specific data from log files and those analyzes (i.e., statistical computations) required to perform all the system control and maintenance for the correct administration of the system and to improve it in terms of performance and security. Moreover, it will manage the resources of the collaborative workspace, which can be managed by a group member acting as an administrator within the group.
- **CSCL knowledge management component**: this manages all the specific and large user events in order to handle the data user interaction as crucial information for the extraction of the essential knowledge to notify users of what is going on in the system as well as to monitor user behavior and control system resources. In order to achieve these goals, this component has been split into the CSCL activity management and CSCL knowledge processing subsystems. The former aims to collect and classify the user events captured according to generic group activity parameters (i.e., task performance, group functioning and scaffolding—see Caballé and Xhafa, 2009, for a detailed description). The latter is responsible for the performance of the statistical analysis of the event information previously handled and contains those statistical criteria which are most common in these environments (e.g., the number of students connected over a period of time, the average student working session). Furthermore, it will enable log information to be exported and extracted in different formats for later statistical analysis in external statistical packages. The final objective of this component is to extract valuable information from the events generated with the aim of revealing useful knowledge.
- **CSCL functionality component**: this forms, along with the previous component, the basis of the collaborative learning environments by defining the three basic elements involved in any groupware application (see Fig. 1) namely, coordination, communication and collaboration. Due to their importance, this component pro-

---

2 Last release of the Collaborative Learning Purpose Library is version 1.1, which can be found at: http://clpl.uoc.es/docs/CLPLdevelopment.zip (Web page as of May 2010).
vides several subsystems or modules so as to provide direct support to each of these areas, namely CSCL coordination, CSCL communication and CSCL collaboration. The coordination support module offers the basic tools to facilitate group organization in planning and accomplishing the members’ objectives as well as group monitoring by modeling the awareness of its participants.

The communication support module involves basic communication elements (Ochoa et al., 2002), which can be implemented in several ways depending on the means of message transmission (point-to-point, multicast and broadcast). Moreover, each message can be delivered asynchronously or synchronously. Finally, the collaboration support module lets members share...
both software and hardware resources in both synchronous and asynchronous.

- The **CSCL functionality** component also supports the presentation of the information (to be collected and processed by the component **CSCL knowledge management**) by means of a subsystem called **CSCL awareness** with the aim of providing participants with immediate awareness of what is going on in the group. Furthermore, in the last few years, feedback is receiving a lot of attention due to its positive impact in online collaborative learning in such areas as group motivation, interaction, or problem-solving abilities (Zumbach et al., 2003). This characteristic is also supported in this component by another subsystem called **CSCL feedback**, which also takes advantage of the knowledge extracted from the group activity to provide participants with a constant flow of as much feedback as possible.

These five CLPL components can be directly reused in the construction of specific efficient, robust, multipurpose and reusable CSCL environments. In addition, important decisions were made as for (i) the incorporation of user and group models; (ii) a generic user interface to allow for graphical and textual modes; (iii) a generic persistence by a technology-independent conceptual data model as part of the PIM, and (iv) the provision of robustness through a complete guidelines that led the development of this platform.

So far, the PIM of the CLPL has been described to model a generic, reusable approach of the CSCL domain. Next sub-section deals with the provision of technology to the PIM in order to achieve the PSM.

### 2.7. Software technology for systematically engineering CSCL applications

Following the principles for GP and MDA developments, once the five components forming the PIM of the CLPL have been fully analyzed and designed, they are to be realized using specific technologies. Two different PSM have been then constructed so far from the PIM: an object-oriented (OO) approach by means of a Java implementation and an approach that follows the service-oriented architecture (SOA) principles. Both technological implementations are described next and their use is justified for the realization of the CLPL, especially from the GP standpoint.

#### 2.7.1. The feasibility of Java for the construction of reusable CSCL software

A great amount of popular learning management systems (LMS) that support collaborative learning have appeared in the marketplace over the last decade, such as Moodle, Blackboard, LRN and Sakai (Bri et al., 2008). Some of them have leveraged the power of the Java programming language to develop robust, portable, scalable, extensible and secure applications. The industry standard of Java is Enterprise Edition (Java EE), which is a widely used platform for server programming in the Java programming language. JEE adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server. Resulting applications are hardware and operating system independent and just rely on having Java installed on the user’s computer. These features allow educational institutions to highly customize e-learning applications to suit their pedagogical needs, and technological requirements.

Along with the positive aspects of JEE, a fairly amount of inconveniences are well-known by the CSCL software development community. The main concern is the poor integration capabilities with legacy and external collaborative learning systems, since the integration solutions should be Java programs and follow the guidelines set by the platform. In addition, the achievement of interoperability often needs time and in-depth knowledge. Other drawbacks of this technology are the steep learning curve, lack of Java expertise in IT departments of educational institutions, and more complexity, larger cost, longer to develop and harder to maintain in comparison to scripting languages.

Current examples of Java-based efforts for collaborative learning purposes are Belvedere, WikiClassroom and TextWeaver. Belvedere (http://belvedere.sourceforge.net) is designed to help support problem-based collaborative learning scenarios with concept and evidence models, and provides multiple representational views (tables and graphs) on those models. It was originally intended to help secondary school students learn critical inquiry skills that they can apply in everyday life as well as in science, but can be adopted to other applications as well. ClassroomWiki (http://cse.unl.edu/~knobel/classroomwiki) is a collaborative writing tool that provides a typical Wikipedia-like interface to the students where he or she can collaborate with his or her group members to create a Wiki on a topic specified by the teacher. This tool incorporates two critical aspects of collaborative learning that typically are not adequately addressed by Wikis: group formation and individual assessment. Finally, TextWeaver (http://sourceforge.net/projects/textweaver/) is a discussion management application to support collaborative learning online designed with specific pedagogical practices in mind. It allows users to read, compose, organize, annotate, and reuse discussion items and related files flexibly and interactively, both online and offline, is a prototype of a new class of discussion management software designed specifically for education.

In addition to the previous assumptions, in order to select Java to build the first PSM of the CLPL, we also assumed the great predisposition of this language to the adaptation and correct transmission of generic software design (see Caballé and Xhafa, 2003, and also Fig. 4). Reusability of the CLPL components in Java is encouraged by...
designing the PIM requirements separately with OO methodology. In order to maintain intact the ideas of GP design that are found, an implicit logical layer is implemented that creates a correspondence between the GP and OO design (see Fig. 4).

Therefore, by codifying the PIM of the CLPL in Java the main objectives of GP and Java’s characteristics were matched:

- Reusability and extensibility allow software to be adapted to many interrelated problems, which is the main aim of GP. Java has many mechanisms such as object type and interface and abstract class which make the CLPL fully susceptible to reutilization. The independence of the platform makes this skeleton portable to most known environments.
- The great potential for the reutilization of GP makes it necessary to guarantee a level of maximum quality. Java has a powerful mechanism of exception management which increases the robustness of the library and hence its quality.
- The Javadoc documentation provided by Java also increases quality by facilitating the test phase and maintenance. GP aims to create software which is as general as possible without losing efficiency by finding the most abstract form of software.
- The simplicity of Java allows the programmer to concentrate on the mechanics of specialization without having to control minor details. Applications with strong user interaction, such as the library, minimize both the relative decrease in performance due to Java being interpreted and the penalization for the casting on use of object.
- The increase in productivity is obtained by the reutilization of existing components. Java has large stores of highly reusable useful code (data structures, etc.) that allows code to be written better and faster and so clearly favoring increased productivity. This Java-based PSM is faithful to this idea.
- Once generic software based on GP has been built, it is then necessary to personalize it to a subgroup of particular requirements so that a specific use within an iterative cycle of abstraction/personalization can be made of it. Due to Java’s capacity, it is feasible to personalize the components of a generic library such as the CLPL components in different ways.

As a result, the Java-based PSM is made up of five packages which constitute the skeleton of the basic structure of whatever application of this domain is constructed using this PSM.\(^4\)

2.7.2. On the advantages of using service-oriented architectures for CSCL

There are a great deal of similarities between collaborative learning needs and benefits provided by SOA (see Section 2 for further information on SOA). As a result of this matching, SOA appears to be an interesting choice to support the development of the most pervasive and challenging collaborative learning environments (Caballé, 2007; Fang and Sing, 2009). In the CSCL context, SOA enhances educational organizations by increasing the flexibility of their pedagogical strategies, which can be continuously adapted, adjusted, and personalized to each specific target learning group. Moreover, SOA facilitates the reutilization of successful collaborative learning experiences and makes it possible for the collaborative learning participants to easily adapt and integrate their current best practices and existing well-known learning tools into new learning goals.

Therefore, in order to increase flexibility and interoperability in CSCL, SOA represents an ideal context to support and take advantage of both the latest trends of software development and the benefits provided by distributed systems for the demanding requirements of the CSCL applications to be completely satisfied. Using SOA in the context of the CLPL offers the following key advantages (Caballé, 2008a):

- Simplifies the encapsulation mechanism that is necessary behind a common interface of diverse implementations;
- Adapts CSCL applications to changing technologies;
- Easily integrates CSCL applications with legacy learning systems and tools;
- Updates pedagogical models and learning tools without causing repercussions on the underlying learning systems and platforms;
- Quickly and easily create and update a learning process from existing services.

From this service-oriented view, current popular LMS platforms, such as Moodle (Al-Ajlan and Zedan, 2007), are evolving by exposing certain aspects of their functionality externally and separating content from tools. However, current LMS still focus strongly on learning administration rather than on the learner (Dagger et al., 2007) and are unable to offer the expected degree of flexibility, interoperability, scalability and reuse as the key challenges for the next generation of the LMS platforms.

Among the efforts that are currently being made by different consortia and initiatives around the world there exist a variety of frameworks, specifications, and guidelines for service-oriented e-learning platforms (Dagger et al., 2007): The IMS Abstract Framework (www.imsglobal.org/specifications.html) loosely identifies and represents the core components and interfaces of an e-learning system. The E-Learning Framework (ELF, www.elframework.org) illustrates e-learning systems’ common functionalities. Similarly, the open knowledge initiative (OKI: www.okiproject.org) defines service layers for developing e-learning platforms. The common approach among these emergent standards is to modularize functionality, usually defining the following groupings (Caballé, 2007): application services (educational domain dependent that provide the functionality required by agents, such as content management and assessment); common services (provide lower-level functionality upon which educational-domain services and users depend, such as authentication, archiving and service directory); core infrastructure services (cross-domain set of services, such as lookup, policy, replica management and scheduler).

Over the last few years, we are witnessing important efforts of collaborative learning tools based on the SOA approach are: Gridcole (Bote-Lorenzo et al., 2008) is a tailable system in which the use of IMS-LD-based collaboration scripts and grid service-based tools is combined by educators in order to support the realization of scripted collaborative learning situations. Elgg (http://www.elgg.org/) is an open source collaborative platform that powers all kinds of social environments for education, from a campus wide social network for universities, schools or colleges. It exposes functionality through the REST API by building a plug-in and either publishes the API for other developers to build clients, or provides an own. Discussion forum (Caballé and Xhafa, 2009) is a common interface of diverse implementations;

4 The Javadoc documentation and source code of the Java-based PSM of the CLPL is found at http://clpl.uoc.edu and http://clpl.uoc.edu/src/clpl-java.zip.

5 Both the WSDL files and the Web-services of this entire CLPL’s PSM are found at http://clpl.uoc.edu/src/clpl-wsdl.zip.
Fig. 5. Excerpt of a WSDL file as an example of the service-oriented PSM.

approach. These standards represent a suitable context to guarantee, reusability, interoperability and scalability. This results in a collection of WSDL files organized in directories that are automatically turned into generic, functional Web-services implemented in the desired programming language and allowing developers to implement these services according to specific needs (Caballé et al., 2007; see also Fig. 5).

Although this PSM was based on SOAP technologies and the adoption of REST for CSCL was non-existent at the time when the CLPL was conceived (Caballé et al., 2004), our CLPL follows certain key principles of REST (see Section 2.3 for a description of the REST architectural style) by prioritizing simplicity, flexibility and scalability while leveraging SOAP/WS-* for meeting complex non-functional requirements when needed, such as security, reliability and transactions. Please note that although the Web-service PSM of the CLPL is strictly implemented with SOAP technologies the specific design makes it possible for this platform to become as simple and flexible as the REST principles claim.

The ultimately aim of the Web-service technological focus of the CLPL aims at supporting as many CSCL applications as possible, whether they are simple or complex by combining both REST and SOAP principles. For simple applications, such as a collaborative Wiki and forum, developers can leverage the CLPL’s Web-services directly in loosely coupled fashion and allow others to consume these services in a REST style (i.e., just developing the appropriate CLPL Web-service for a particular purpose and publishing it on the Web to be consumed by clients). Please note that clients are to be prepared to consume any CLPL Web-service by wrapping up them with the appropriate Web-service environment corresponding to the client’s programming language. On the other hand, in order to face complexity, such as the construction of a new virtual campus, REST technologies lack the needed security, reliability, and transactability of SOAP. By leveraging its SOAP orientation, the CLPL can potentially meet these and other non-functional complex requirements by means of incorporating the SOAP/WS-* specifications.

It is worth noticing that no complex Web-services specifications, in particular SOAP/WS-*, have been incorporated into the CLPL so far in order to keep the approach as simple and flexible as possible. In particular, the general purpose and application domain specific management interfaces addressed by WS-*, such as WS-ResourceProperties and WS-notifications, are addressed by the CLPL in an ad hoc way. Thus, for instance, the CLPL provides the subsystem CSCLSessionManagement to manage ad hoc the logics of the state representation of the CLPL resources in a programmatic way. Notifications are also addressed by this subsystem by allowing each and every Web-service interface of the CLPL to explicitly request the resources’ value change to this subsystem.

To sum up, the combination of MDA, SOA, and Web-services results in a PSM as a collection of WSDL files organized in directories. They are automatically turned into generic Web-services by Apache Axis, allowing developers to implement the services according to specific needs and using the most appropriate language.

The ultimate aim of the CLPL is to enable a complete and effective reutilization of its generic components and subsystems in a simple and flexible manner as the skeleton for the construction of concrete CSCL applications. Thus, this platform implements the conceptualization of the fundamental needs existing in potentially most of collaborative learning experiences. In addition, the CLPL is highly interoperable in distributed environments permitting complete flexibility of the services offered in terms of implementation languages and underlying software and hardware platforms.

3. Methodology

This section presents a methodological approach to validate the previous software platform to develop CSCL applications. First, an application example is briefly described in the form of a new interactive collaborative learning tool to support the online discussion processes happening in the virtual classrooms of the Open University of Catalonia (UOC). This application is then justified from the intensive use of the CLPL platform to build it and as basis for final project’s students to extend this application and achieve effective and timely developments of CSCL systems. Finally, the statistical models used for the elaboration on the data collected from the experiments are described.

3.1. An application example: a structured discussion forum

To illustrate the use of the CLPL platform presented in Section 2, a prototype of a Web-based application called discussion forum (DF) (Caballé and Xhafa, 2009) was developed to validate the possibilities offered by this platform during the construction of new software to support collaborative learning in online environments.

---

6 Apache Axis forms part of the Apache Project, found at http://apache.org/axis (Web page as of May 2010).
7 The UOC is located in Barcelona, Spain and can be found online at http://www.uoc.edu (Web page as of May 2010). Since 1994 the UOC offers distance education through the Internet. About 52,000 students, lecturers and tutors participate in some of the 600 on-line official courses available from 23 official degrees and other PhD and post-graduate programs.
8 See http://clpl.uoc.edu/df for reaching the portal of the discussion forum (as of May 2010).
An implementation prototype of this application was built through the extensive use of the CLPL platform described above with the aim of giving new opportunities to learning methodologies, such as learning by discussion (see Figs. 6 and 7), and being applied to new learning scenarios. The DF provides significant benefits for students in the context of project-based learning, and in education in general. Please see Caballé and Xhafa (2009), for a complete description of this application.

Taking advantage of the flexibility of the service-oriented approach, we used different languages for the development of a prototype of the DF for both the client and the server sides. Thus, on the one hand, PHP resulted in a very suitable programming language to implement the web pages forming the user interface on the client side. On the other hand, the generic Web-services supporting the business and data layers on the server side were implemented in Java as a powerful and experienced language featuring very well as to robustness, portability, ease of use and extensibility, which created an ideal context for the implementation on the server side.

The real context of this tool is the virtual learning environment of the UOC, which offers higher education over the Internet. Given the added value of asynchronous discussion groups, the UOC have...
incorporated online discussions as one of the pillars of its pedagogical model. Therefore, great efforts are being made to develop adequate online tools to support the essential aspects of the discussion process, which include students’ monitoring and evaluation.

This prototype is currently working as a typical client-server Web-based application at the Open University of Catalonia and evolving rapidly to be completed. Several online courses are using this tool to support their discussion processes as part of the very rationale of their pedagogical goals. As a result, a total of more than 700 graduate and undergraduate students from three courses in computer science have been involved directly or indirectly in collaborative learning experiences by using the DF (see Caballé and Xhafa, 2009, for a full description and the results achieved from these experiences).

For the purposes of this research, the ultimate goal of this application is to set the grounds of the evaluation process of the CLPL platform in terms of effectiveness, quality and development time of new software in the domain.

### 3.2. Adopted analyses and procedures for data elaboration

Master and bachelors thesis courses offered by the UOC dispose specific areas related to software engineering and in particular the development of software applications for collaborative learning. The interested areas are called “Web-based applications for collaborative work” and “Computer-Supported Collaborative Learning”. These courses are intended to provide the needed resources and framework in support for students who develop collaborative tools for e-learning by exclusively using the CLPL platform as a primary resource.

Master and bachelors thesis courses’ curricula base part of the success of the learning process on allowing students to deploy and test their own CSCL applications with real users onto the servers of the virtual campus. Because of the particular learning model and technological constraints of the UOC, the virtual campus support asynchronous collaborative learning only. In addition, among other technological constraints, Web 2.0 tools for collaborative learning are slowly incorporated. Therefore, students of these courses are encouraged to develop those types of CSCL applications that can be empirically validated in the virtual campus. Indeed, we believe that those types of unsupported collaborative learning are not fully representative and form a little fraction of our unique model of collaborative learning, whose lack do not restrain at all the day-to-day learning processes as confirmed by the great success of the UOC’s pedagogical model.

The ultimate goal of these courses is to extend the structured DF presented in the previous sub-section with representative, complete, autonomous CSCL applications (such as a collaborative agenda and calendar, file repository for group work, and a voting system for collaborative discussions), which each forms entire software development projects to be fully completed within a 14-week course. Considering the course time is rather short and the novelty for students to develop real complex software projects, the use of the CLPL becomes a major resource to fulfill the task and pass the course.

The main support provided from the course is twofold, the lecturer’s guidance during the whole development, and the organization of the course’s curricula into a few deliverables that students are required to submit in deadline fashion. These deliverables are planned to fit the different phases of the traditional software development process (i.e., specification, design, and implementation) plus both an initial stage to plan and organize the whole project and a thesis’ defense at the end of the course.

From the beginning (i.e., early this decade), a great deal of graduate and undergraduate students of the UOC have chosen these interest areas to develop their thesis. The courses involved in these areas were quite demanding in terms of time and efforts to develop and deploy a full software application in a short time. As a result, many students dropped out the courses because they could not fulfill certain goals of the courses’ curricula, such as the submission of required deliverables in time and the low quality of the developments. The teaching staff decided then to alleviate the course load by incorporating technical documentation exemplifying similar software developments performed in previous courses as well as standard code libraries. This novelty provided students with an initial reuse capability though very poor and informal and as a result the benefits in terms of reusability were also very little (see Table 1 on the productivity row with standard resources).

#### 3.2.1. Quantitative and qualitative evaluation procedures

Two different approaches are combined to analyze the data collected from the experiments performed by using both the Standard resources offered by the course (e.g., past developments repositories, code libraries, etc.) and, in addition, the CLPL platform for developing new software in the real learning context of the Open University of Catalonia. The benefits from using both resources are compared and evaluated.

In particular, a first approach is a quantitative evaluation which involves the identification of the number of diagrammatic artifacts reused when modeling in UML the stages of specification and design of the software application in hand, including the amount of code reused in the implementation phase. For instance, Fig. 8 shows how the UML diagram of the business model of the CLPL is reused and particularized into the business model of an electronic voting system (EVS) for CSCL purposes. In addition, the number of deliverables submitted in time and their qualification were also considered. A typical thesis course such as those mentioned in this study, involves the submission of three partial deliverables and a final deliverable that summarizes the entire thesis work. However, as aforementioned, for the purposes of this study it

### Table 1

Variables of interest and data collected from using the standard and the CLPL resources for 4 representative projects.

<table>
<thead>
<tr>
<th>Variable of interest</th>
<th>Resources</th>
<th>Projects</th>
<th>Total (on average)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User and group management</td>
<td>Agenda and calendar</td>
<td>Electronic voting system</td>
</tr>
<tr>
<td>#Productivity (reused diagrams in %)</td>
<td>Standard</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>CLPL</td>
<td>90</td>
<td>75</td>
</tr>
<tr>
<td>#Effectiveness (timely deliverables in %)</td>
<td>Standard</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>CLPL</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td>#Quality (assessment by instructor on average, scale 0–10)</td>
<td>Standard</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>CLPL</td>
<td>8.0</td>
<td>8.0</td>
</tr>
</tbody>
</table>
was not essential that the students involved in the experiences reported passed the course since we considered every deliverable individually and we extracted the percentage of timely submitted deliverables. To sum up, the variables of interest are: the increase of productivity (i.e., UML artifacts reused), effectiveness (i.e., development time in terms of timely deliverables submitted) and quality (i.e., work assessment by the instructor). The aim is to evaluate the level of guidance and support for students of the CLPL through the different stages of the software development in comparison to the use of the standard resources offered by the course.

The second approach is a qualitative evaluation which was addressed to students to share their experiences when using the CLPL for developing new software. The students were asked to fill out and submit a questionnaire reporting on their degree of satisfaction, confidence and motivation during the course by focusing on the CLPL as a course’s primary resource.

4. Results and discussions

In this section the results achieved are shown and then discussed on the benefits and problems from using the CLPL platform for developing new software for meeting collaborative learning needs.

4.1. Quantitative results

Both types of resources (i.e., CLPL platform and standard software resources) are considered for the derived variables: (a) number of UML diagrams and other artifacts reused from other sources, (b) number of deliverables submitted in time during the course, and (c) average final marks achieved for the whole course. Both types of resources offered by the course are involved in all experiments (i.e., projects) and thus are used to collect the data. Table 1 shows the results for the variables of interest considered, namely productivity, effectiveness, and quality. Finally, our experience by using the CLPL shows a high level of reusability, which reaches about 70% on average.

The results in Table 1 lead us to formulate and discuss on the following statements regarding the variables of interest:

- Students showed a dramatic increase on productivity when using the CLPL platform. Although by using other standard software resources (e.g., documentation repositories from previous courses) slightly increased the productivity in comparison to earlier courses, the great reusability potential of the CLPL caused a dramatic increase in production since its incorporation into the courses. In addition, considering implementation is usually the only benefited stage from software reusability (Czarnecki
Table 2
Excerpt of a questionnaire’s results on the use of the CLPL platform.

<table>
<thead>
<tr>
<th>Selected questions</th>
<th>Average of responses (0–5)</th>
<th>Excerpt of students’ comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assess in general the CLPL to help develop new software</td>
<td>4</td>
<td>“Despite the learning curve is high I became very productive and could finish my project in time. I could reuse not only code but specially modelling diagrams.”</td>
</tr>
<tr>
<td>Evaluate the increase of productivity by using the CLPL</td>
<td>5</td>
<td>“It was impressive to develop complex software in such a short time (…) and with high marks!”</td>
</tr>
<tr>
<td>Evaluate how the CLPL impacted on the development time</td>
<td>5</td>
<td>“Just copying the diagrams I could make progress fast and without making errors. It would be great if importing and editing the diagrams into my modelling tool.”</td>
</tr>
<tr>
<td>Evaluate how the CLPL improved the quality of your software</td>
<td>4</td>
<td>“After two academic terms failing the course due to lack of time I could submit the deliverables and the final project in time. I wish I could have used the CLPL before”</td>
</tr>
<tr>
<td>Compare the CLPL with the standard software resources offered in the course</td>
<td>5</td>
<td>“I plan to use the CLPL in my future developments, some components are generic enough to serve in other domains. The saving of time and efforts is immense!”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“I got a good knowledge on software development. I think reuse-based courses should be incorporated at UOC and the CLPL used in basic courses of software development.”</td>
</tr>
</tbody>
</table>

and Eisenicker, 2000), the CLPL mainly provides great reusability capabilities on the early stages of the software development by reusing modeling artifacts during the specification and design stages. For instance, use case, class and collaboration diagrams were just copied as such from the CLPL and particularized into the specific needs (see Fig. 8). This procedure guarantees neither to oversights neither important aspect nor make simple modeling mistakes, and instead gives clear and correct guidelines to lead all stages of development. In overall, the impact is much greater than just simply reusing code.

• The increase of effectiveness is also significant when comparing the use of the CLPL to previous experiences with standard software libraries and development repositories. Almost twice as many deliverables were submitted in time as previous experiences. Indeed, by reusing many modeling and code artifacts as such from the CLPL, students speeded up their work and were capable of submitting the courses’ deliverables in time or at worst case with a slight delay upon the course’s schedule. In addition to saving time, by the high level of reusability achieved students produced more exhaustive and detailed developments and of more quality.

• The last variable of interest, quality, was also significantly improved from previous CLPL experiences. Final marks assessing the whole software project developments increased about 30%. As seen in Section 2, quality is another main repercussion when reusing at large scale. Indeed, by reusing well experimented pieces of software, the resulting new software inherits a high and increasing level of correctness and robustness, which provides the required degree of software reliability. During both the software modeling and the implementation stages, the resulting students’ deliverables were implicitly correct in those parts that were fully reused from the CLPL. Just the particularization processes as well as new development forming the nuclear requirements of the project’s developments (i.e., less than 30% of the total development) showed certain level of inaccuracy.

To sum up, by reusing the UML diagrams and other modeling artifacts from the CLPL during the specification and design phases of their developments, students became more productive, saving time and efforts without being worried about the quality of the reused material, which was guaranteed to be high. The implementation phase was also largely benefited from reusing the code skeleton generated by the CLPL (either as Java or Web-services PSM).

4.2. Qualitative results

Table 2 shows an extract of the results of the questionnaire addressed to 20 students who participated in the experience in five academic terms. They were asked about their experience of using the CLPL.

From the qualitative results obtained from these questionnaires, students show a high degree of satisfaction, confidence and motivation when extensively using the CLPL in their software developments. As a result, the identification of the requirements and their analysis and design by reusing the UML diagrams of the CLPL were highly satisfactory and of good quality. Similar effects are found when extensively reusing any PSM of the CLPL during the implementation stages.

In particular, students reported saving time and effort by avoiding starting from scratch but having 70% on average of the development already fulfilled instead. Most importantly, they reported on feeling highly confident in developing the applications since the CLPL provided them with strong guidance and support in terms of going through the different stages of the software development and the UML modeling at any stage. Indeed, the idea of copying existing modeling diagrams and other artifacts rather than creating them from scratch fosters students to go on their developments. Diagrams are then particularized in a simple manner (see Fig. 8), which saves a lot of time and efforts while keeping quality high by inheriting well tested material.

On the other hand, students reported on having to overcome a high learning curve when first facing the CLPL documentation and procedures. A reason may be found on the lack of previous courses focusing on essential issues of software reutilization, such as the GP paradigm. In addition, students reported on having problems to take on the great amount of technological issues imposed by the CLPL (e.g., SOA and Web-services approach). However, these issues were largely compensated for reusing at large scale and eventually students benefited from the CLPL as shown in Table 1.

From the instructors’ perspective, the CLPL approach also benefited them by bringing a systematic way to monitor and assess the students’ deliverables. Instructors reported that this software platform alleviates them from the tedious work of paying attention on the details of the common parts of the developments (70% on average). Instead, they rely on the CLPL experience and evaluate on the reuse degree achieved. Indeed, the CLPL’s reuse capabilities become maximum degree when working on the common requirements (e.g., user management, authentication and authorization, system administration, etc.). Therefore, instructors concentrate just on the specific aspects of the developments (30% on average).

In overall, these results are not conclusive but they encourage us to undertake more experimentation and especially validation processes on the large scale reuse possibilities provided by the CLPL platform.

5. Conclusions and further developments

This paper proposes a step further in the current software development methodologies by taking advantage of the most advance and latest techniques in software engineering, such as GP and SOA. The goal is to greatly improve software development in terms of
quality, productivity and timely developments, as well as providing effective solutions to meet demanding and changing requirements.

To this end, an architectural solution in the form of a generic, highly reusable software infrastructure called CLPL has been presented and evaluated to help develop complex, flexible, and advanced collaborative learning applications.

Both the development experience of the CLPL and of a specific application, called DF, based on this platform, are reported to validate the key ideas proposed in this contribution. In addition, more validation process is provided by reporting the use of this software platform as the primary resource for master’s thesis students to develop new software in the CSCL domain. From the main results extracted after analyzing the experiences achieved in different courses for about 10 academic terms, we conclude that the CLPL platform is a promising effort towards the timely and effective development of CSCL applications of high quality.

Despite encouraging, these results are not conclusive due to the exploratory nature of the approach. More experiences are expected to prove and validate the CLPL as the primary platform to support students’ thesis at the UOC when developing complex and demanding applications for collaborative learning.

Following students’ suggestions, ongoing work is to make the CLPL’s modeling artifacts importable into current modeling tools in order to avoid rewriting them. This is indeed an improvement that we plan to offer shortly by considering XMI files (see OMG, 2006 for details), which are XML-tagged files as the result of coding UML diagrams, so that the CLPL’s PIM can be editable on different modeling tools and thus can save even more time and effort by avoiding to draw them in the designer’s favorite modeling tool. Lack of comply with standards of the existing UML case tools is the major problem to face next. Then, by combining XMI technology with XSL style sheets it is possible to turn the PIM’s XMI files into WSDL files, which represent the input for a Web-service working environment to transform them into a specific-language architecture design (i.e., PSM).

Following this procedure, we plan to automatically describe WSDL files from the PIM model so that it is possible to generate PSM implementations of the CLPL in different programming languages.

Finally, since the CLPL is based on SOAP, we plan to adopt the intrinsic features of REST and incorporate them into the CLPL by redesigning and implementing the whole platform. The ultimate aim is to provide a hybrid approach to make it possible to select the appropriate technology for each and every CSCL application developed with the CLPL.

Acknowledgements

This work has been partially supported by the European Commission under the Collaborative Project Alice “Adaptive Learning via Intuitive/Interactive, Collaborative and Emotional System”, VII Framework Programme, Theme ICT-2009.4.2 (Technology-Enhanced Learning), Grant Agreement n. 257639, and the Spanish MCYT project COLE “Contributory Learning systems”, TIN2008-01288/TSG. Fatos Xhafa’s work is partially done at Birchbeck, University of London, on leave from Technical University of Catalonia (Barcelona, Spain). His research is supported by a grant from the General Secretariat of Universities of the Ministry of Education, Spain.

References


Santi Caballé has a PhD, masters and bachelors in computer science from the Open University of Catalonia. Since 2003, he has been an assistant professor at the Open University of Catalonia teaching a variety of courses in computer science in the areas of information systems, software engineering and collaborative learning. Since early 2006 he has been working as an associate professor of the department of computer science, multimedia and telecommunication at the Open University of Catalonia where he coordinates several online courses in the area of software engineering. His research focuses on e-learning, software engineering, network technologies, distributed learning, computer-supported collaborative learning, learning interaction analysis, and grid technologies. His home page is located at http://cv.uoc.edu/~scaballe.

Fatos Xhafa received his PhD in computer science from the Polytechnic University of Catalonia (Barcelona, Spain) in 1998. He joined the Department of Languages and Informatics Systems of the Polytechnic University of Catalonia as an assistant professor in 1996 and is currently an associate professor. He is member of the ALBCOM Research Group of this department. His research is supported by several research projects from Spain, European Union and NSF/USA. He has published in leading international journals and conferences. He serves in the editorial board of several leading journals and has served as co-chair/PC member for many conferences and workshops.