

CLPL: Providing Software Infrastructure for the Systematic and Effective Construction of Complex Collaborative Learning Systems

***Abstract*— 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 proposes a innovative approach 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 propose 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 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 in the CSCL domain.

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

***Index Terms*—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.**

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 (Ateyeh 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.

To this end, a generic, robust, flexible, interoperable, reusable computational model that meets the fundamental functional needs shared by any collaborative learning experience 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 on-line 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 (Pahl, 2007) with the aim of reusing the large number of common requirements shared by CSCL applications. 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 & Greenberg, 1996). In addition, the representation and analysis of group 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 (Martínez, de la Fuente and Dimitriadis, 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 so 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, 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 (Czarneki, 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" (Ateyeh 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 et al., 2004).

This contribution proposes a 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. To this end, we propose an advanced reuse-based software engineering methodology for developing CSCL applications in an effective and timely fashion. A validation process of the effects of this approach is provided by the on-line 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 on-line collaborative learning experience as well as to achieve a more effective collaboration (McGrath, 1991; MacDonald, 2003; 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; Stalh, 2006; Daradoumis et al., 2006). Collaborative or group learning refers to instructional methods 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 & 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 & 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.

- Efficiency. It is also essential to guarantee the efficiency of components as if this not done the performance repercussions will be noted, just as with lack of quality, in all of the systems involved.
- Productivity. Inherent to reutilization is the saving through not having to create software components again that already exist. Hence, there is an increase in computing production.
- Automation. The aim is to automate the processes so that general requirements with a high level of abstraction and specially designed tools can be used to produce operative programmes.
- Personalisation. As the general requirements are made more particular, so the product that is generated becomes more optimised to meet the specific needs of the client.

GP also represents one important technique to achieve effective Product Lines (PL) following the Product-Line Architecture(PLA) approach (Gomma, 2004). PLA promotes developing large families of related software applications quickly and cheaply from reusable components. In PLA, a certain level of automation is provided in the form of generators (also known as component configuration tools) to realize solutions for large parts of the systems being developed.

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 behaviour 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 organisations sup-ported by heterogeneous hardware and software platforms to share and use the same services,
- facilitate integration of other existing systems and thus protect previous in-vestments (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 realised 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 organisations (e.g., W3C consortium).

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 (OMG, 2006): service-oriented architectures (SOA) and product line architectures (PLA). As to the former, SOA provides great flexibility to system architectures by organizing the system as a collection of encapsulated services. Hence, SOA relies on services which represent the behavior

provided by a component to be met and used by any other components based only on the interface contract. As to the latter, PLA promotes developing large families of related software applications quickly and cheaply from reusable components.

There are many views and opinions about what MDA is and is not. However, the OMG, as the most authoritative view, focuses MDA on a central vision (OMG, 2006): Allow developers to express applications independently of specific implementation platforms (such as a given programming language or middleware). To this end, OMG proposes the following principles for MDA developments: first, the development of a UML-based Platform Independent Model (PIM), second, one or several models which are Platform Specific Models (PSM). Finally, a certain degree of automation by means of descriptions is necessary for mapping from PIM to PSM.

2.5. A generic gaze at the collaborative learning applications

In this section, a generic view of the CSCL domain is given by analyzing and taking into account the commonality found in the requirements of most of collaborative learning environments.

In the last years there has been an explosion of new CSCL applications aiming to create 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, the collaborative applications must provide support to three essential aspects: coordination, collaboration and communication; with communication being the base for reaching coordination and collaboration (Roseman & Greenberg, 1996). Collaboration and

communication might be *synchronous* or *asynchronous*. The former (Stahl, 2006) means cooperation at the same time with typically fine-grained notifications giving immediate feedback about the activities of other participants whereby the shared resource (such as a text document and a message) will not have a lifespan beyond the sharing. The latter (Dillenbourg, 1999b) means cooperation at different times and the shared resource will be stored in a persistent support.

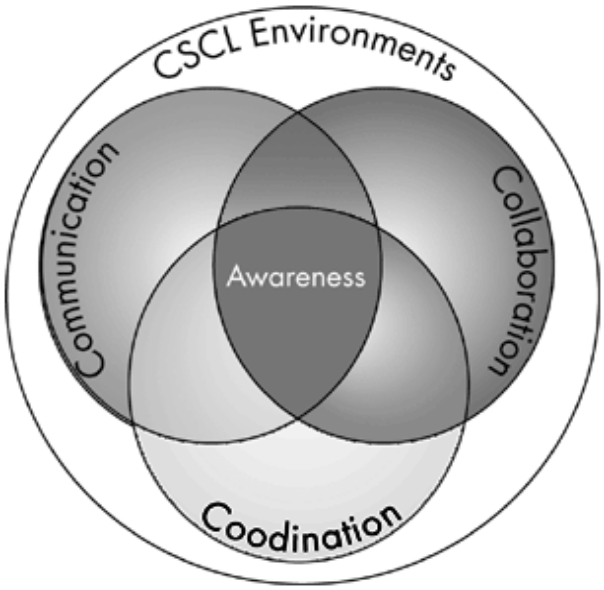


Figure 1. The essential aspects in any collaborative learning (CSCL) application.

The different areas overlap each other (see Figure 1) and any collaborative system must support all of them (Ochoa et al., 2002):

- Coordination is an important aspect of any collaborative activity. It entails the combination and sequencing of otherwise independent work toward the accomplishment of a larger goal (McGrath, 1991). In a collaborative learning environment, coordination mostly refers to the tasks toward the learning group formation and the definition and

planning of the group objectives. Moreover, the group coordinator may track task status, deadlines, resource usage, working results, or other critical process parameters to correctly lead the group.

- Collaboration relies on students sharing all kind of documents. The sharing of resources between several participants is therefore a central functionality of CSCL systems (Stahl, 2006). Sharing may be synchronous, with several participants accessing the same resource at the same time (that is, they work on the same copy of the document), or asynchronous, with different participants accessing the same resource at different times (each of them works on a different copy of the same document).
- Communication is another functional aspect of collaboration systems aiming to support the communication between two or more collaborative learning participants (Baloian, 2002). Communication includes text messages, spoken interactions, or non-verbal exchanges like gestures in a video conference. Communication may take place asynchronously (different participants communicate at different times such as email, debate, etc.) or synchronously (participants communicate at the same time such as chat, video conference, etc.). The communication support is based on four elements involved: a message as the information carrier between a sender process and a recipient process (which receives and possibly process the message) through a channel (Ochoa et al., 2002). Moreover, in this context, it is necessary to implement different ways of message addressing such as point-to-point, multicast and broadcast.
- Awareness (Gutwin et al., 1995) is essential for any of the three forms of cooperation seen above. It allows for implicit coordination of collaborative learning, opportunities for informal, spontaneous communication and gives users the necessary *feedback*

(Zumback et al., 2003) about what is happening in the system. In particular, on the one hand, synchronous awareness lets users know exactly what other co-participants are doing (e.g. during a shared editing session shows who is editing what) and when documents are in use by others. On the other hand, asynchronous awareness determines who, when, how and where shared resources have been created, changed or read by others.

In order to improve the collaboration within a group it is important to take into account both current and future behavior of all user types and the fact that user objectives or intentions may change as they interact with the system. To that end, it is essential to design some kind of user and group models describing, for example, the user characteristics, intentions, beliefs, knowledge, skills, roles and collaborative activities (Brusilovski, 1996). Moreover, the user and group models should be open enough to let add new services and collaborative activities to them according to the participant needs.

The design of the CSCL user interface offers many more challenges than the design of interfaces for single user applications (e.g. multi-user editors). The user interface must provide information about what others are doing to efficiently support collaborative tasks and additional information has to be presented. The latter refers to the effects of other users' activities which must be communicated by visual or audio signals. Therefore, the user interface is the main way to support awareness in multi-user collaborative environments.

Finally, although most research efforts in CSCL areas have been dedicated to developing distance learning environments, most learning activities still take place in the traditional face-

to-face classroom (Baloian et al., 2002). To that end, the generic approach presented in this paper should support the common basis from both scenarios and it is possible to instantiate CSCL applications both for virtual learning (i.e. most of participants are physically in different places) and for traditional learning (i.e. all the participants are physically found in the same place, usually in a classroom). In this paper, though it mostly refer to virtual CSCL environments, the principles are the same for both scenarios.

2.6. Software infrastructure for CSCL applications

The main contribution of this paper is a generic, reusable, robust, flexible, interoperable, component-based and service-oriented platform called Collaborative Learning Purpose Library (CLPL)[1] (Caballé et al., 2007).

The CLPL is based on the Generic Programming paradigm so as to enable a complete and effective reutilization of its generic components as the skeleton for the construction of any collaborative learning application. This generic platform implements the conceptualization of the fundamental needs existing in any collaborative learning experience.

In order to meet these requirements, the development of the CLPL is based on the Model-Driven Development (MDD) paradigm and the framework supporting it, namely Model-Driven Architecture (MDA) (Czarnecki, 2005). In proposing MDA, the CLPL development takes advantage of two key ideas that have had significant influence in addressing the current

[1] Last release of the CLPL is version 1.1, which can be found at: http://clpl.uoc.edu/docs/CLPLdevelopment.zip (Web page as of November 2009).

challenges in software development (Caballé, 2008): Service-Oriented Architectures (SOA) and Product Line Architectures (PLA). As to the former, SOA provides great flexibility to system architectures by organizing the system as a collection of encapsulated services.

Hence, SOA relies on services which represent the behavior provided by a component to be met and used by any other components based only on the interface contract. As to the latter, PLA promotes developing large families of related software applications quickly and cheaply from reusable components. In PLA, a certain level of automation is provided in the form of generators (also known as component configuration tools) to realize solutions for large parts of the systems being developed (Czarnecki, 2005). Taking these approaches into consideration, the CLPL is based on SOA and the Generic Programming paradigm (Czarnecki, and Eisenecker, 2000; Caballé, and Xhafa, 2003) as the central part of the development in MDD.

In particular, in developing the CLPL, a Platform Independent Model (PIM) was first created by applying the following Generic Programming 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 Figure 2). Second, two different Platform Specific Model (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.

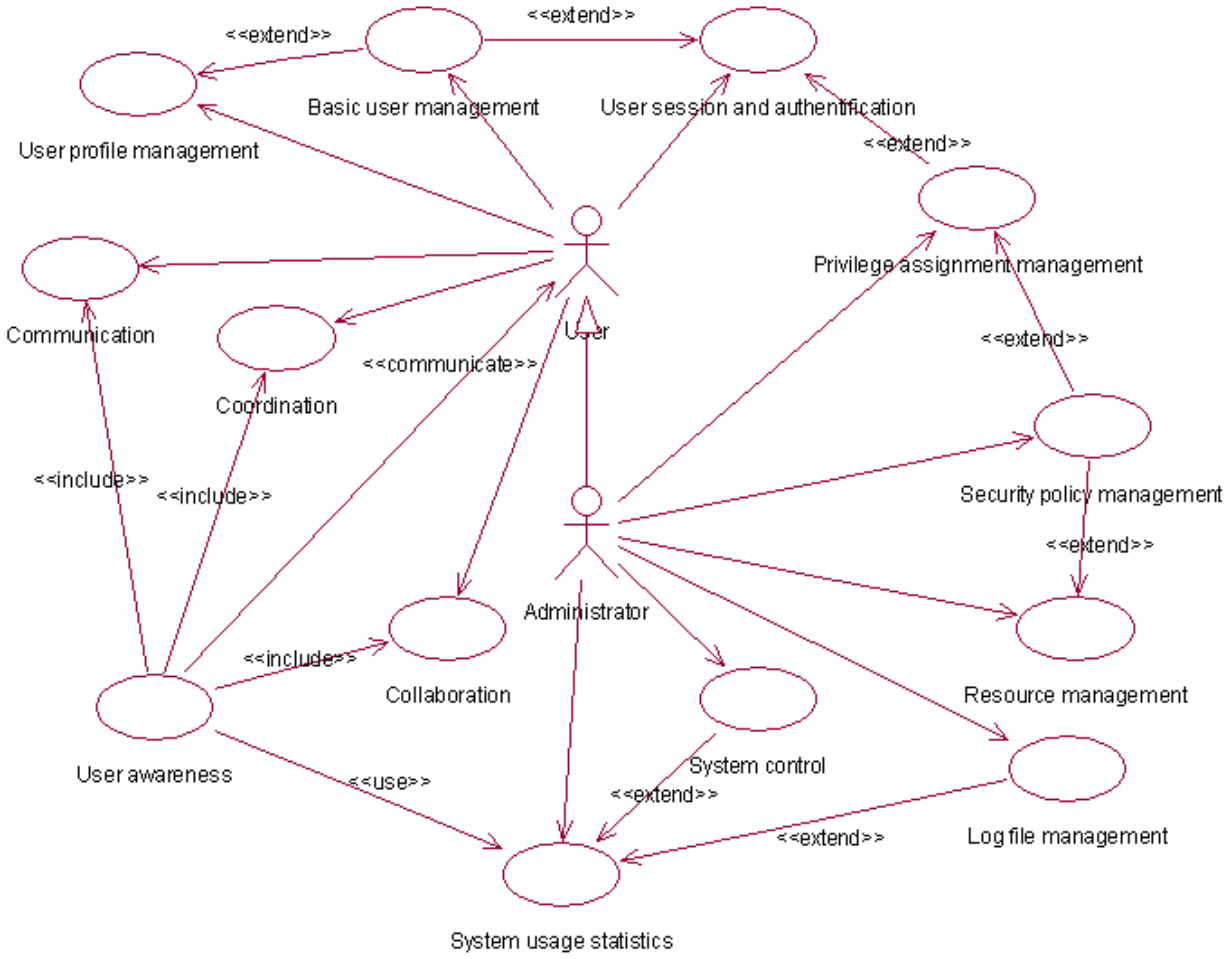


Figure 2: UML-based use case diagram with the general requirements of the CLPL.

The ultimate aim of the CLPL is to enable a complete and effective reutilization of its generic services and components as the skeleton for the construction of any collaborative learning application, and in particular CSCL applications. Thus, this platform implements the

conceptualization of the fundamental needs existing in any collaborative learning experience. 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.

For the rest of this section an UML-based PIM model for the CLPL is described by means of a general view of the CLPL architecture[2]. Next sub section faces the PSM approach by incorporating specific technology to the CLPL.

2.6.1. The CLPL architecture

The CLPL (Caballé et al., 2007) is made up of five components (see Figure 3) handling user management, administration, security, knowledge management, and functionality, which map the essential issues involved in any collaborative learning application.

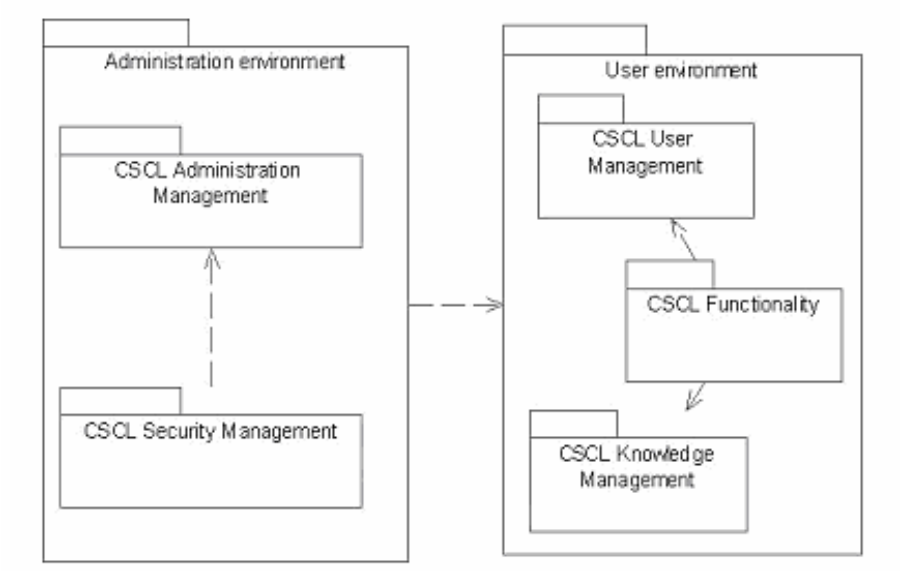


Figure 3. Graphical representation of the CLPL components.

[2] The complete PIM of the CLPL is found at http://clpl.uoc.edu/docs/CLPLdevelopment.pdf

- *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, thus this component provides the generic *ProfileElement* entity which dynamically allows new user and group needs to be met.
- *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. Its genericity lets programmers implement these issues with the latest cryptographic security mechanisms.
- *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.

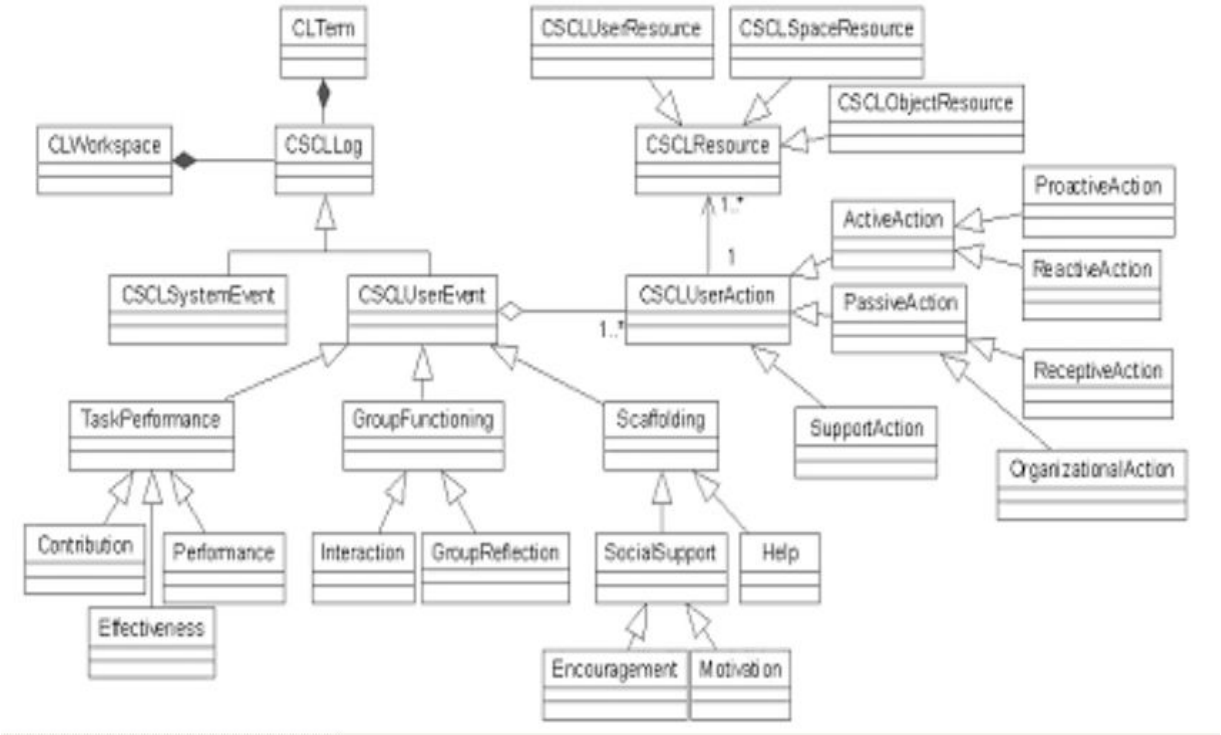


Figure 4. A class diagram to collect and classify all events generated during the group activity.

- *CSCL Knowledge Management* component: this manages all the specific and large user events in order to handle the data of 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. To this end, 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 a complete hierarchy of user events (see Fig. 4) provided, which is based on the above-mentioned three generic group activity parameters: task performance, group functioning (i.e., interaction behavior) and scaffolding. The latter is responsible for the

performance of the statistical analysis of the event information previously handled and includes another generic hierarchy (see Fig. 5) that 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.

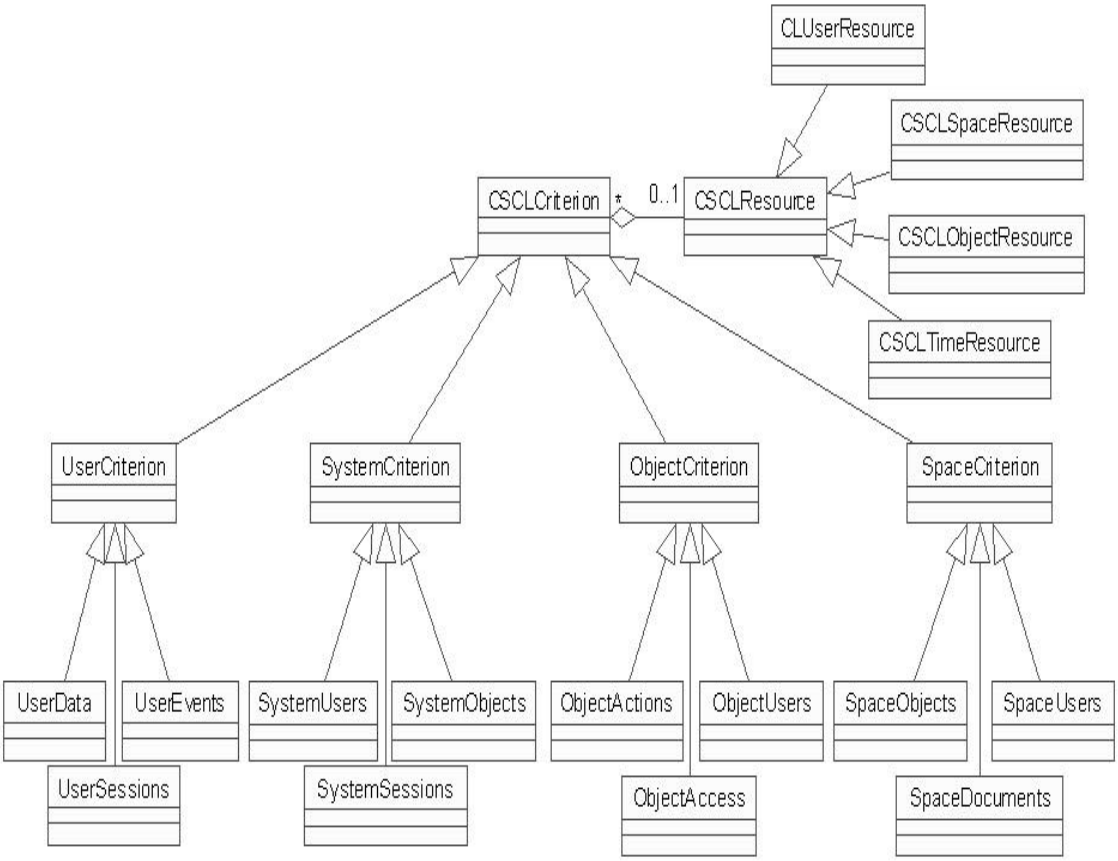


Figure 5. A hierarchy to classify criteria in Web-based applications.

- *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 Figure 1) namely, coordination, communication and collaboration. The different areas overlap each other, and any collaborative system must support all three aspects. Due to their importance, this component provides several subsystems or modules so as to provide direct support to each of these areas, namely *CSCL Coordination*, *CSCL Communication* and *CSCL Collaboration* (see Figures 6 and 7). 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 four basic elements, the *sender*, *message*, *channel* and *receiver* (Ochoa, 2002), and 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 (as in the case of an email, where the message is made persistent by default) or synchronously (as in a chat, where conversation is made persistent so that it can later be processed). Finally, the collaboration support module lets members share both software and hardware resources in both synchronous (e.g. real-time editors) and asynchronous (e.g. file sharing) modes.

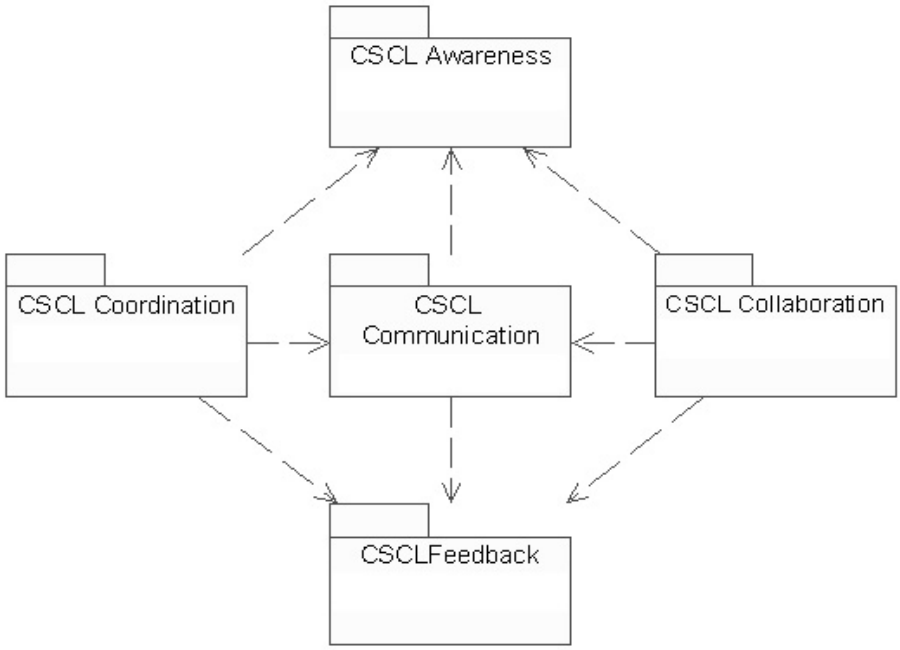


Figure 6. Graphical representation of the subsystems making up the *CSCL Functionality* component.

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* (see Figure 7) 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 on-line collaborative learning in such areas as group motivation, interaction, or problem-solving abilities (Zumbach, at al, 2003). This characteristic is also supported in this component by another subsystem called *CSCL Feedback* (see Figure 7), 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.

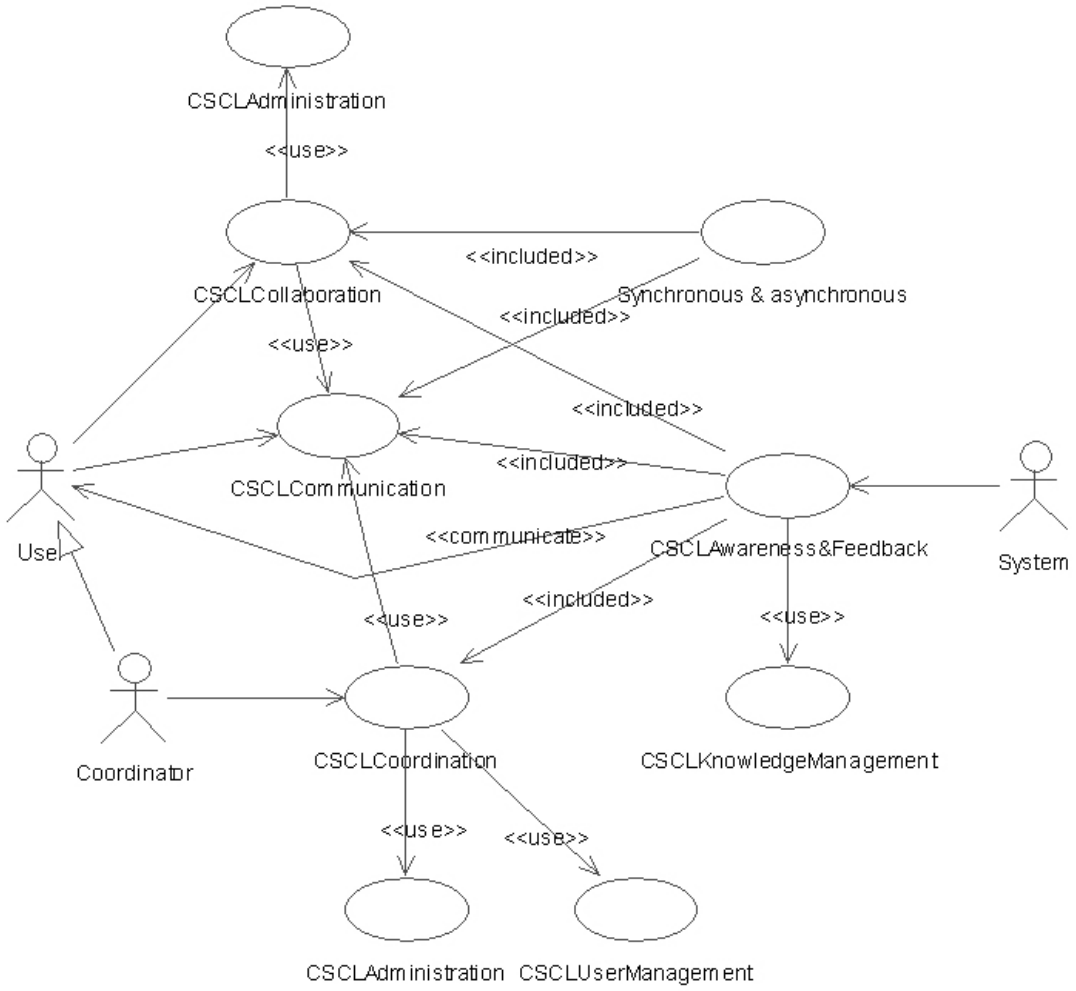


Figure 7. The analysis of the *CSCL Functionality* component.

These CLPL components can be directly reused in the construction of specific efficient, robust, multiplatform and reusable CSCL environments. The following are important decisions made and guidelines that led the development of the CLPL:

- In order to improve collaboration within a group, it is important to take into account both current and future behavior of all user types and the possibly changing objectives and intentions of the users as they interact with the system. To this end, generic user and

group models have been designed to describe the users' characteristics, intentions, beliefs, knowledge, skills, roles and collaborative activities amongst others. Moreover, the user and group models are sufficiently open as to allow new services and collaborative activities to be added in accordance with the needs of the participants.

- The design of the user interface in collaborative learning applications offers many more challenges than the design of interfaces for single user applications (e.g. multi-user editors). The user interface must provide information about what others are doing to efficiently support collaborative tasks, and awareness information regarding the effects of other users' activities has to be communicated by visual or audio signals. The user interface is therefore the main way to support awareness in multi-user collaborative environments. Furthermore, the user interface is generically focused so as to make particularization in graphical and text modes possible. Even though the user interface in collaborative learning environments will usually be in graphic mode, it is necessary to consider generic focusing in order to make the logic part of the application independent from the specific design of the graphic user interface.

- The design of the persistence in the CLPL is also generic and thus a disk manager abstraction has been considered. The disk manager acts as a bridge between the future application and its data to make the design of the persistence independent from the specific technology that will manage the data. This way, it is possible to treat both ordinary text files and different database system managers during particularization. Furthermore, a complete technology-independent conceptual data model is provided as

part of the PIM (see Figure 8), which may be realized in different technologies managing generic persistence.

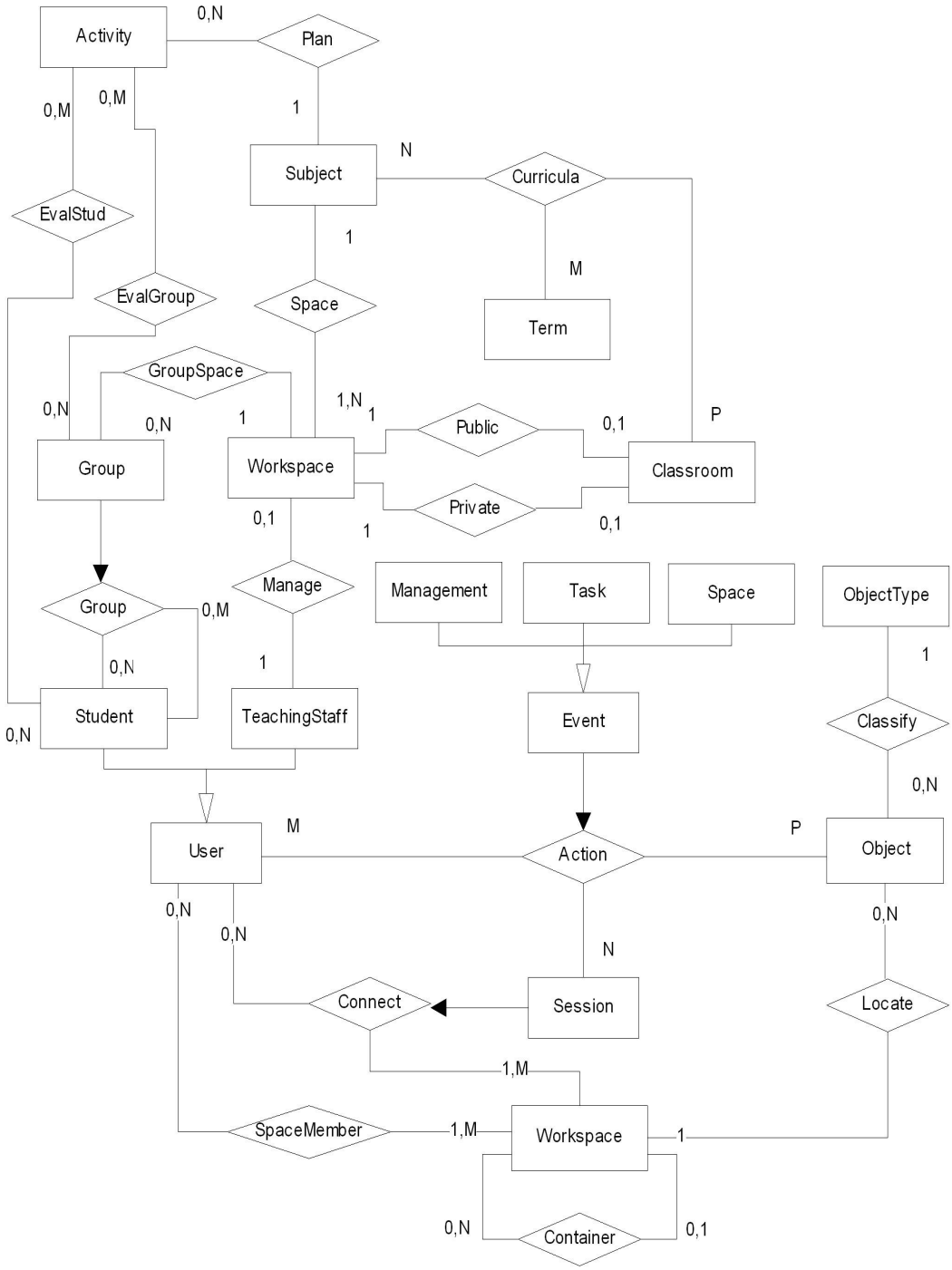


Figure 8. ER diagram representing the conceptual design of the database.

- Robustness is offered through a complete hierarchy of error treatment and so a high degree of the component quality and reliability is guaranteed without depending on the error treatment of the specific platform supporting the software.

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

2.7. Software technology for systematically engineering CSCL applications

Following the principles for GP and MDA developments, once these five components forming the PIM of the CLPL have been fully analyzed and designed, they are to be realized using specific technologies. To this end, two different Platform Specific Model (PSM) have been 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 technology approaches 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

The first PSM of the CLPL is based on the Java programming language due to its great predisposition to the adaptation and correct transmission of generic software design (see Caballé and Xhafa, 2003 and an example in Figure 9). To this end, in order to encourage the reusability of the CLPL components the basic requirements forming the PIM are designed

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. 9).

```
package clpl.users;

import gpl.users.*;
/**
 *  Abstract class carrying out the logic part of the group creation within a CSCL environment.<br>
 *  @author <a href="mailto:scaballe@uoc.edu">Santi Caballé Llobet</a><br>
 *  @see <a href="http://cv.uoc.edu/~scaballe/gpl/api/gpl/users/GenericUserCreation/">gpl.users.GenericUserCreation</a><br>
 *  @version 1.0a; 15-12-2003; java version "1.2.2"<br>
 */
public abstract class CLGroupCreation extends GenericUserCreation{

    /**
     *  Constructor without parameters from the class CLGroupCreation.<br>
     *  Creates a group.<br>
     *  (pre: the group does not exist in the data store.)<br>
     *  (post: the group has been created in the data store.)<br>
     */
    public CLGroupCreation() throws GenericUserException, Exception {
    }
}
```

Figure 9. An example of the Java-based PSM of CLPL as coded design

In 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 adapt 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 specialisation 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 specialize the components of a generic library such as the CLPL components in different ways.

As a result this 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[3].

[3] 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.

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

The second PSM of the CLPL was developed following the principles of Service-Oriented Architecture (SOA) and realized using Web-services (Caballé, 2008).

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 the best choice to support the development of the most pervasive and challenging collaborative learning environments. 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, the second PSM of the CLPL relies on SOA as it 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 et al., 2008):

- Simplifies the encapsulation mechanism that is necessary behind a common interface of diverse implementations
- Adapts CSCL applications to changing technologies.

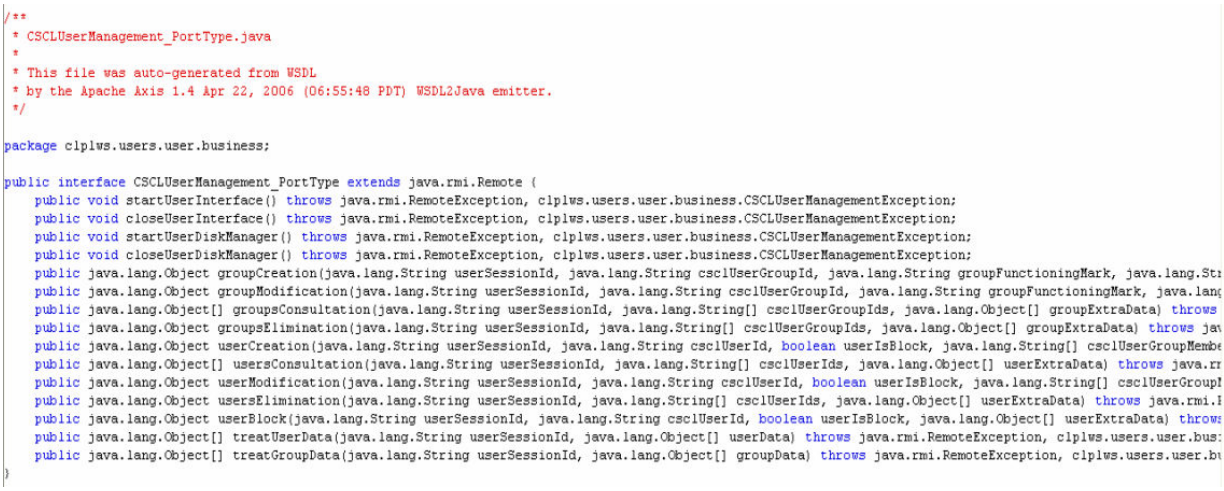
- Easily integrates CSCL applications with legacy learning systems and tools.
- Updates pedagogical models and learning tools without causing repercussions on the underlying learning systems and platforms.
- Quickly and easily create and update a learning process from existing services.

Web-services were the implementation technology chosen for this CLPL's PSM[4] given the widely adopted protocols and standards, which represents the rationale of the Web-services approach. These standards represent a suitable context to guarantee interoperability and scalability by taking great advantage of the distributed technologies. 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 Figures 10 and 11).

```
<wsdl:message name="groupCreationRequest">
   <wsdl:part name="userSessionId" type="soapenc:string"/>
   <wsdl:part name="csclUserGroupId" type="soapenc:string"/>
   <wsdl:part name="groupFunctioningMark" type="soapenc:string"/>
   <wsdl:part name="groupOutcomes" type="soapenc:string"/>
   <wsdl:part name="groupType" type="soapenc:string"/>
   <wsdl:part name="csclUserCoordinatorId" type="soapenc:string"/>
   <wsdl:part name="csclUserTutorId" type="soapenc:string"/>
   <wsdl:part name="csclUserGroupMembersIds" type="impl:ArrayOf_soapenc_string"/>
   <wsdl:part name="csclWorkspaceId" type="soapenc:string"/>
   <wsdl:part name="groupExtraData" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
```

Figure 10. Excerpt of a WSDL file as an example of the service-oriented PSM.

[4] 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



```
/**
 * CSCLUserManagement_PortType.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis 1.4 Apr 22, 2006 (06:55:48 PDT) WSDL2Java emitter.
 */

package clplws.users.user.business;

public interface CSCLUserManagement_PortType extends java.rmi.Remote {
    public void startUserInterface() throws java.rmi.RemoteException, clplws.users.user.business.CSCLUserManagementException;
    public void closeUserInterface() throws java.rmi.RemoteException, clplws.users.user.business.CSCLUserManagementException;
    public void startUserDiskManager() throws java.rmi.RemoteException, clplws.users.user.business.CSCLUserManagementException;
    public void closeUserDiskManager() throws java.rmi.RemoteException, clplws.users.user.business.CSCLUserManagementException;
    public java.lang.Object groupCreation(java.lang.String userSessionId, java.lang.String csclUserGroupId, java.lang.String groupFunctioningMark, java.lang.Str
    public java.lang.Object groupModification(java.lang.String userSessionId, java.lang.String csclUserGroupId, java.lang.String groupFunctioningMark, java.lang
    public java.lang.Object[] groupsConsultation(java.lang.String userSessionId, java.lang.String[] csclUserGroupIds, java.lang.Object[] groupExtraData) throws
    public java.lang.Object groupsElimination(java.lang.String userSessionId, java.lang.String[] csclUserGroupIds, java.lang.Object[] groupExtraData) throws ja
    public java.lang.Object userCreation(java.lang.String userSessionId, java.lang.String csclUserId, boolean userIsBlock, java.lang.String[] csclUserGroupMembe
    public java.lang.Object[] usersConsultation(java.lang.String userSessionId, java.lang.String[] csclUserIds, java.lang.Object[] userExtraData) throws java.rr
    public java.lang.Object userModification(java.lang.String userSessionId, java.lang.String csclUserId, boolean userIsBlock, java.lang.String[] csclUserGroupI
    public java.lang.Object usersElimination(java.lang.String userSessionId, java.lang.String[] csclUserIds, java.lang.Object[] userExtraData) throws java.rmi.I
    public java.lang.Object userBlock(java.lang.String userSessionId, java.lang.String csclUserId, boolean userIsBlock, java.lang.Object[] userExtraData) throws
    public java.lang.Object[] treatUserData(java.lang.String userSessionId, java.lang.Object[] userData) throws java.rmi.RemoteException, clplws.users.user.busi
    public java.lang.Object[] treatGroupData(java.lang.String userSessionId, java.lang.Object[] groupData) throws java.rmi.RemoteException, clplws.users.user.bu
}
```

Figure 11. An example of a Web-service generated in a specific programming language.

To sum up, the combination of MDA, SOA, and Web-services results in a platform-specific model (PSM) as a collection of WSDL files organized in directories. They are automatically turned into generic web-services by Apache Axis[5], 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 services and components as the skeleton for the construction of any collaborative learning application, and in particular CSCL applications. Thus, this platform implements the conceptualization of the fundamental needs existing in any collaborative learning experience. 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. To this end, first, an application example is shown 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[6]. This application is then described from the intensive use of the CLPL platform to built it and as basis for final project's students to extend this application and achive 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 structured discussion forum was developed[7] to validate the possibilities offered by this platform during the construction of new software to support collaborative learning in online environments. Therefore, the objective of this sub section is to show a representative example of the development of a Web-based structured discussion forum called Discussion Forum (DF) (see Caballé and Fatos, 2009, for a complete description of this application) and provide an implementation prototype of this application through the extensive use of the CLPL platform described above to support the discussion process. The ultimate goal is to

---

[5] Apache Axis forms part of the Apache Project, found at http://apache.org/axis (Web page as of November 2009).

[6] The Open University of Catalonia (UOC) is located in Barcelona, Spain. The UOC offers distance education through the Internet since 1994. About 50,000 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. The UOC is found at http://www.uoc.edu (Web page as of November 2009).

[7] see http://clpl.uoc.edu/df for reaching the portal of the Discussion Forum application.

set the grounds of the evaluation process of this platform in terms of effectiveness, quality and development time of new software in the domain.

To this end, we first describe the pedagogical requirements of the DF and then provide the main guidelines that conducted the development of a prototype of this application that gives new opportunities to learning methodologies, such as learning by discussion, and be applied to new learning scenarios (Caballé and Xhafa, 2009). This application provides significant benefits for students in the context of project-based learning, and in education in general.

3.1.1. Pedagogical background and requirements

In collaborative learning environments, the discussion process forms an important social task where participants can think about the activity being performed, collaborate with each other through the exchange of ideas that may arise, propose new resolution mechanisms, and justify and refine their own contributions and thus acquire new knowledge (Stahl, 2006).

To this end, a complete discussion and reasoning process is proposed based on three types of generic contributions, namely *specification*, *elaboration* and *consensus*. *Specification* occurs during the initial stage of the process carried out by the tutor or group coordinator who contributes by defining the group activity and its objectives (i.e. statement of the problem) and the way to structure the group activity in sub-activities. *Elaboration* refers to the contributions of participants (mostly students) in which a proposal, idea or plan to reach a solution is presented. The other participants can elaborate on this proposal through different types of participation such as questions, comments, explanations and

agree/disagree statements. Finally, when a correct proposal of solution is achieved, the *consensus* contributions take part for its approval (this includes different consensus models such as voting); when a solution is accepted the discussion terminates.

In a discussion process, participants perform a role according to their profile (e.g. coordinator, member, guest, etc.), have personal collaborative preferences (e.g. language) and must set up environment features (e.g. sound or visual effects, text or voice warnings, etc.) according to their personal characteristics. Participant needs are not static and they evolve as the discussion moves forward.

3.1.2. The design of the application

During the design of this application, the interesting features and requirements mentioned above were supported by allowing the application to take advantage of the CLPL components. Representative correspondences are described here.

The *CSCL Functionality* component provided suitable support in the design of the virtual places where the discussions take place. For instance, the *room* entity was recursively used in different levels of abstractions, such as folders to hold the debate, and discussion threads inside each debate (see Fig. 13-a). This also eased the implementation by reusing the same code for both purposes. Another important purpose of this component was to support the provision of feedback to users from the interaction data collected and analysed. (see Fig. 13-b).

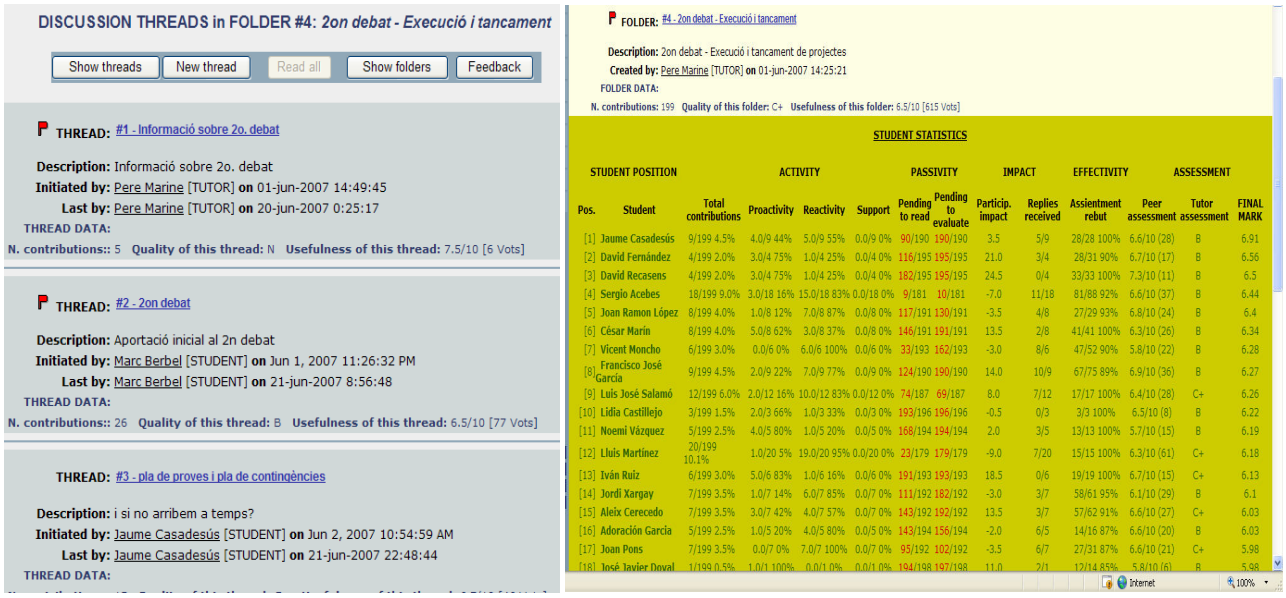


Figure 13. Two snapshots of the DF prototype: a) A list of discussion threads; b) an example of the provision of complex feedback to participants

In designing the specification phase, coordination needs are to be supported by essential elements such as an agenda and a calendar so as to perform all the typical tasks in this initial stage of the discussion process (such as group formation, definition of objectives, structuring the task in sub-activities and labor division). During this phase the *CSCL Coordination* subsystem provided support by certain generic entities that were particularized into specific needs of this application. In order to enable the tutor to both monitor and assess the discussion process, the application took advantage of the generic report system provided by this subsystem so as to keep track of the performance of participants and assess their contributions.

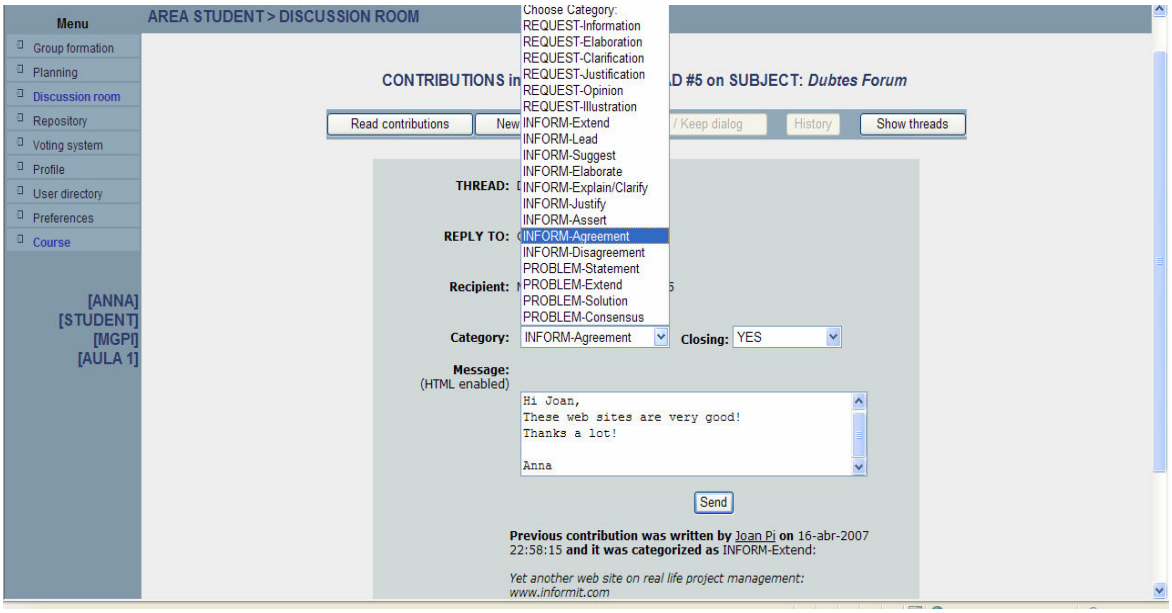


Figure 14. A list of tags to qualify a contribution.

The application design includes certain thematic annotation cards (such as question, idea, response, etc.) that structure the elaboration phase and can offer full help support as well (see Fig. 14). All events generated are recorded as user actions, analyzed and presented as information to participants either in real time (to guide directly students during the learning activity) or after the task is over (in order to understand the collaborative process).

To that end, the *CSCL Knowledge Management* component provided full support to event management. In particular, during the elaboration phase, a complete treatment of the structured task performance events generated enables the system to keep participants aware of the contributing behavior of others, to check certain argumentative structures during the discussion and also to open up the possibility to provide feedback based on the data produced (see Fig. 13-b). Equally, group analysis outcomes produced by the treatment of group functioning events constitute an important data source that can assist in achieving a more satisfactory solution to the problem during the consensus phase. Furthermore, the

coordinator can use this same information to organize well-balanced groups during the specification phase.

Personal features of the discussion group participants (their role, collaboration preferences and so on) were taken into account and a user and group model were designed so as to allow participants to add new services whilst their needs evolve as the discussion moves forward. All these user features were included by the *CSCL User Management* component through the user profile management subsystem, providing a solid support for building and maintaining the user and group model.

Therefore, on the one hand, the structured discussion forum supports a complete discussion process through the realization of three generic contribution types and an open user and group model. On the other hand, this application constitutes a valuable resource that takes advantage of the computational model to greatly improve essential features of a discussion process such as awareness of participant contributions and enhance the abilities of users by increasing their knowledge of each other in terms of motivation, interaction behavior and so on.

3.1.3. Implementation and exploitation of the tool

Taking advantage of the flexibility of the service-oriented approach, we used different languages for the development of a prototype of the structured discussion forum 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 Open University of Catalonia (UOC) , which offers higher education over the Internet. Given the added value of asynchronous discussion groups, the UOC have incorporated on-line discussions as one of the pillars of its pedagogical model. Therefore, great efforts are being made to develop adequate on-line tools to support the essential aspects of the discussion process, which include students' monitoring and evaluation.

To this end, 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 this tool (see Caballé and Xhafa, 2009, for a full description and the results achieved from these experiences).

3.2. Adopted analyses 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.

The ultimate goal of these courses is to extend the structured discussion forum (DF) presented in the previous sub section with complete, autonomous new functionality (such as a collaborative agenda and calendar, and a voting system) and forming 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 two-fold, 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 the real learning context in a short time. As a result, many students dropped out the courses because they could not fulfill 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 row on *productivity* with Standard resources).

Since the Fall term of 2004, and for 10 academic terms so far, all graduate and undergraduate students of the UOC that have chosen these interest areas are required to use the CLPL as the main course's resource to perform their software developments. Despite some students still drop out because of personal reasons[8], they can perform part of work. Representative efforts are four applications, namely a group and user manager, file repository, collaborative agenda and calendar - all of them intended to support the personal and group management and work - as well as an electronic voting system to support the consensus part in the discussion process[9]. Each student is required to intensively use and reuse the CLPL as much as possible from the very first step of the project development. In addition, students can still make the most of the technical documentation with similar developments existing in the course's repository from the very beginning of these courses.

[8] Because of the particular profile of the UOC (students are about 30 years old on average and 95% with a job), the dropout ratio is about 50%.

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 diagramatic 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. In addition, the number of deliverables submitted in time and their qualification were also considered. To sum up, the variables of interest are: the increase of productivity (i.e., number of 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. To this end, 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 primary resource.

[9] An example of a deliverable based on the CLPL for an electronic voting system (EVS) application for is found at: http://clpl.uoc.edu/docs/EVS_Specification_Nov2009.pdf. Please note that student's

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.

Variable of interest	*Resources*	*Projects*				*Total (on average)*
		User & group management	*Agenda & Calendar*	*Electronic voting system*	*File repository*	
# Productivity (reused diagrams in %)	Standard	20	5	5	10	10
	CLPL	90	75	60	85	77.5
# Effectiveness (timely deliverables in %)	Standard	60	50	30	40	45
	CLPL	90	80	80	80	82.5
# Quality (assessment by instructor on average, scale 0-10)	Standard	6.5	6.5	5.0	6.5	6.1
	CLPL	8.0	8.0	7.5	8.0	7.8

Table 1. Variables of interest and data collected from using the Standard and the CLPL resources for 4 representative projects.

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.

---

personal data has been removed not to disclose the anonymity.

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 and Eisenecker, 2000), the CLPL also 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. 15). This procedure guarantees not to oversight any important aspect nor make simple modeling mistakes and 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 on 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 software forming the nuclear needs 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 5 shows an extract of the results of the questionnaire addressed to students. They were asked about their experience of using the CLPL.

Selected questions	Average of responses (0 – 5)	Excerpt of students' comments
Assess in general the CLPL to help develop new software	4	"Despite the learning curve is high I became very productive and could finish my project on time. I could reuse not only code but specially modelling diagrams." "It was impressive to develop a complex software in such a short time (…) and with high marks!" "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." "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" "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!"
Evaluate the increase of productivity by using the CLPL	5	
Evaluate how the CLPL impacted on the development time	5	
Evaluate how the CLPL improved the quality of your software	4	
Compare the CLPL with the standard software resources offered in the course	5	

Table 5. Excerpt of a questionnaire's results on the use of the CLPL platform.

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 to start 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. 15), which saves a lot of time and efforts while keeping quality high by inheriting well tested material.

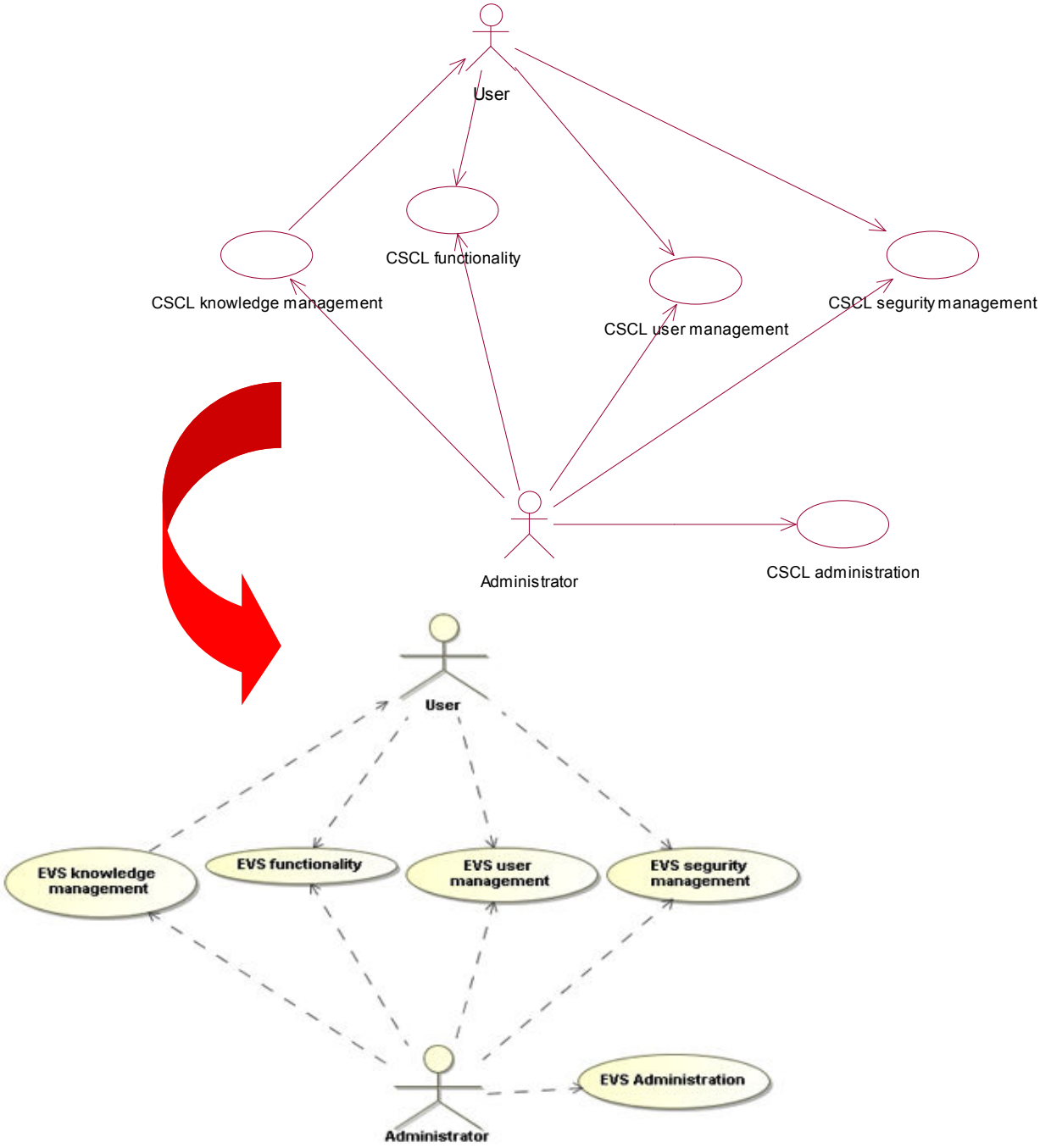


Figure 15. An example of particularization by using the CLPL. The business model of the CLPL is particularized into the business model of an electronic voting system (EVS) for CSCL purposes. Both diagrams are the same and just the names in the general use cases are particularized. Please note that both diagrams were drawn with different modeling tools.

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 generic programming 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 proves to work on the common parts where it is most reused (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 Generic Programming and Service-Oriented Architectures. The goal is to greatly improve software development in terms of quality, productivity and timely developments, as well to provide 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 to help develop complex, modern and advanced collaborative learning applications.

Both the development experience of the CLPL and of a specific application, called Discussion Forum, 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 come and validate the CLPL as the *de facto* platform to support students' thesis at he 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. To this end, the latest research results are leading us to deal with 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 any modeling tool 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 standard of the existing UML case tools is the major problem to face next.

Finally, 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 (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.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish MCYT project TIN2008-01288/TSI. Fatos Xhafa's work is partially done at Birkbeck, 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

Ateveh, K. & Lockemann, P. C. (2006). Reuse- and Aspect-Oriented Courseware Development. Educational Technology & Society, 9 (4), 95-113.

Bacelo Blois, A., Becker, K. (2002) A Component-Based Architechture to Support Collaborative Application Design, In Haake and J. Pino Eds., Groupware: Design, Implementation and Use. LNCS, Vol. 2440, pp. 134-143.

Baloian, N., Galdames, P., Collazos, C., & Guerrero, L. (2004). A model for a Collaborative Recommender System for Multimedia Learning Material. Proceedings of the 10th International Workshop on Groupware. Berlin: Springer.

Brusilovski, P. (1996). Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6(2-3), 87-129.

Caballé, S., & Xhafa, F. (2003). A Study into the Feasibility of Generic Programming for the Construction of Complex Software. Proceedings of the 5th Generative Programming and Component Engineering, Erfurt, Germany. ISBN: 3-9808628-2-8.

Caballé, S., Xhafa, F., Daradoumis, T., & Marquès, J.M. (2004). Towards a Generic Platform for Developing CSCL Applications Using Grid Infrastructure. Proceedings of the First International Workshop on Collaborative Learning Applications of Grid Technology, Chicago, IL, USA.

Caballé, S., Daradoumis, T., & Xhafa, F. (2007). A Generic Platform for the Systematic Construction of Knowledge-based Collaborative Learning Applications. Architecture Solutions for e-Learning Systems. Idea Group Inc (IGI). Chp. XII (pp. 219-242). Press, Hershey, PA: Idea Group, ISBN: 978-1-59904-633-4.

Caballé, S. (2008). Combining Generic Programming and Service-Oriented Architectures for the Effective and Timely Development of Complex e-Learning Systems. Proceedings of CISIS 2008. Barcelona, Spain. IEEE Computer Society.

Caballé, S., Xhafa, F. (2009). Fostering Collaborative Knowledge Building by the Effective Provision of Knowledge about the Discussion Process. International Journal of Business Intelligence and Data Mining (IJBIDM). Vol. 4, No. 2, pp. 141-158. ISSN: 1743-8187. Inderscience Publishers.

Czarnecki, K. Overview of Generative Software Development (2005). In J.-P. Banâtre et al. (Eds.): Unconventional Programming Paradigms (UPP) 2004, Mont Saint-Michel, France, LNCS 3566, pp. 313–328.

Czarnecki, K. and Eisenecker, UW (2000). Generative Programming: Methods, Techniques, and Applications. Addison-Wesley.

Daradoumis, T., Martínez, A., and Xhafa, F. (2006). A Layered Framework for Evaluating Online Collaborative Learning Interactions. International Journal of Human-Computer Studies. Academic Press: Elsevier Ltd.

Dillenbourg, P. (1999a). Introduction; What do you mean by “Collaborative Learning”? P. Dillenbourg (Ed.), Collaborative learning. Cognitive and computational approaches, 1-19. Oxford: Elsevier Science.

Dillenbourg, P. (1999b). Collaborative Learning. Cognitive and Computational Approaches. Elsevier Science Ltd, 1-19.

Gomaa, H. Designing Software Product Lines wtih UML: From Use Cases to Pattern Based Software Architectures. Reading, Massachusetts: Addison-Wesley, 2005.

GuiLing W., YuShun L., ShengWen Y., ChunYu M., Jun Xu, Meilin S., (2005). Service-Oriented Grid Architecture and Middleware Technologies for Collaborative E-Learning. IEEE SCC 2005: 67-74.

Gutwin, C., Stark, G., & Greenberg, S. (1995). Support for Workspace Awareness in Educational Groupware. Proceedings of the ACM Conference on Computer Supported Collaborative Learning, Bloomington, Indiana, USA.

Koschmann, T. (1996). Paradigm shifts and instructional technology. In T. Koschmann (Ed.), CSCL: Theory and Practice of an Emerging Paradigm, Mahwah, New Jersey, Lawrence Erlbaum Associates, (1-23).

Martínez, A., de la Fuente, P., and Dimitriadis, Y. (2003). Towards an XML-based representation of collaborative interaction. B. Watson, S. Ludvigsen, & U. Hoppe (Eds.), Proceedings of the International Conference on Computer Support for Collaborative Learning 2003, Bergen (pp. 379–384). Dordrecht: Kluwer Academic Publishers.

McGrath, J.E. (1991). Time, Interaction and Performance: A Theory of Groups. Small Group Research, 22, 147-174.

Ochoa, S., Guerrero, L.A., Fuller, D. and Herrera, O. (2002) Designing the Communication Infrastructure of Groupware Systems. In Groupware: Design, Implementation, and Use. J. Haake and J. Pino (eds.). Lecture Notes in Computer Sciences, Vol. 2440, Springer Verlag. September, 1002. pp. 114-133.

OMG (2006), MDA specification guide. Version 1.0.1. Report – omg/03-06-01.

Pahl, C. (2007). Architecture Solutions for e-Learning Systems. Hershey, PA, USA: IGI Global. ISBN: 978-1599046334.

Roseman, M. and Greenberg, S. (1996). Building Real Time Groupware with GroupKit, A Groupware Toolkit. March. ACM Transactions on Computer Human Interaction, 3(1), p66-106, ACM PressTeam New York, NY, USA

Sfard, A. (1998). On two metaphors for learning and the dangers of choosing just one. Educational Researcher 27(2) 4-13.

Soller, A. (2001). Supporting Social Interaction in an Intelligent Collaborative Learning System. Int. J. of Artificial Intelligence in Education, 12: 40-62.

Stahl, G. (2006). Group Cognition: Computer Support for Building Collaborative Knowledge. Acting with Technology Series, Cambridge, MA: MIT Press. ISBN: 978-0262195393.

Strijbos, J-W., Martens, R., Prins, F., & Jochems, W. (2006). Content analysis: What are they talking about? Computers & Education. 46(1): 29-48, Academic Press: Elsevier Ltd, January 2006.

Webb, N. (1992). Testing a theoretical model of student interaction and learning in small groups. R. Hertz-Lazarowitz and N. Miller (Eds.), Interaction in Cooperative Groups: The Theoretical Anatomy of Group Learning, 102-119. Cambridge Univ. Press, NY.

W3C Working Group (2004). Web Services Architecture Document.. http://www.w3.org/TR/ws-arch/ (Web page as of November 2009).

Zumbach, J., Hillers, A., & Reimann, P. (2003). Supporting Distributed Problem-Based Learning: The Use of Feedback in Online Learning. T. Roberts (Ed.), Online Collaborative Learning: Theory and Practice, 86-103. Hershey, PA: Idea.