The Journal of Systems and Software 83 (2010) 2083–2097

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

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

Santi Caballé [a,*], Fatos Xhafa [b,1]

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

ARTICLE INFO

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

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

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 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.

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 (Ateveh 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

* Corresponding author. Tel.: +34 93 3263439; fax: +34 93 3568822.
E-mail addresses: scaballe@uoc.edu (S. Caballé), fatos@lsi.upc.edu (F. Xhafa).
[1] Tel.: +34 93 4137880; fax: +34 93 4137833.

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.06.013

the very rationale of all CSCL applications (Martínez 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 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 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 (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" (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, 2008a).

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 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.