Collaborative Learning Patterns: Assisting the Development of Component-Based CSCL Applications

Juan I. Asensio, Yannis A. Dimitriadis, Marta Heredia, Alejandra Martínez, Francisco J. Álvarez, María T. Blasco
University of Valladolid. Spain
{juaase@tel,yannis@tel,mherrod@pireo.tel, amartine@infor,fraalv@pireo.tel,tbq@dlyl}.uva.es

César A. Osuna
Mexican Petroleum Institute. Mexico
cosuna@imp.mx

Abstract

The creation of a framework of software components and their associated software design patterns would provide great benefits for the development of reusable, flexible, and customizable component-based CSCL applications. The development of such a framework implies that software developers have a proper understanding of the key concepts and principles of the domain of interest. The achievement of this understanding is particularly difficult in the CSCL domain, where there is a big separation among abstractions used by Educational Science experts and those used by software developers. In order to alleviate this problem, the paper proposes, justifies, and illustrates the use of the so-called Collaborative Learning Patterns: detailed descriptions of well-accepted types of collaborative learning activities defined by Collaborative Learning experts. We also present the initial steps that would be followed so that software developers identify software components applicable to several types of component-based CSCL applications. All this proposal is illustrated with the jigsaw and pyramid Collaborative Learning Patterns and their use in the development of a real CSCL application according to the Unified Process software development methodology.

1. Introduction

During the last years, the technologies associated to the Component-Based Software Engineering (CBSE) have emerged as a promising solution for the achievement of software reusability, flexibility, and maintainability in the development of complex software systems composed of smaller pieces [11]. The potential benefits of this new software development paradigm became even more evident within the field of Educational Software where the social and pedagogical particularities of each educational context have traditionally been translated into large collections of incompatible and monolithic applications thus obstructing the widespread usage of information technologies in classrooms [18]. This problem is even worse when dealing with a particular type of Educational Software: Computer Supported Collaborative Learning (CSCL) applications. CSCL is an Instructional Technology research paradigm based on socially oriented learning theories [14] that underline the influence of social interactions as key learning mediators [8]. CSCL applications have to include support for collaborative activities as well as to offer the functionality desired by the set of potential actors that can participate in collaborative learning situations (teachers, students, and pedagogy and psychology experts, among others). The effort involved in the development of useful and powerful CSCL applications is only justified if they can be applied to a large number of learning situations and if they can survive the evolution of functional requirements and technological changes [17]. Therefore, CBSE appeared as an enabling technology for the development of reusable, customizable, and integrated CSCL software tools.

When dealing with reusability in Software Engineering, and particularly in CBSE, one concept appears as fundamental: that of Component Frameworks [10]. A Component Framework can be understood as a set of extensible software components usable in a particular domain as well as a collection of software design patterns that document their use [4,6]. Components defined in a Framework can be reused, customized and assembled with additional components provided by developers in order to obtain specific applications. The availability of a Component Framework for the CSCL domain would therefore imply a great step towards the achievement of the aforementioned objectives of reusable, integrated and customizable Collaborative Learning software applications. Nevertheless, building a Component Framework is not an easy task [5]. A Frame-

work developer must face different problems related to both the particularities of the Framework domain and the technologies used to support the derived components [15]. One of the most important problems to take into account in this context is the identification and dimensioning (i.e. level of granularity) of components. The fulfillment of this task largely depends on how the key concepts and principles of the domain of interest are understood by software developers [2]. This is a case where a software engineering problem (component reuse) is largely influenced by a knowledge engineering problem (the understanding of the domain). In the CSCL domain, this problem is particularly important due to the big separation among abstractions used by experts in Collaborative Learning (pedagogues, psychologists, education practitioners,…) and those used by software developers. During the last years the authors, within a multidisciplinary education and technology group, have worked in a top-down conceptualization of the Collaborative Learning domain by defining the educational-telematic framework DELFOS [16] and in a bottom-up approach aimed at the understanding of the domain by receiving feedback from the field trials of several specific CSCL applications developed by the group [9]. Both approaches have not generated completely satisfactory results for the objective of conceptually bridging the Collaborative Learning and the Software Development worlds. As a result, this paper proposes, justifies and illustrates a novel approach to the conceptualization of a part of the Collaborative Learning domain as a first step for obtaining a complete Component Framework for the CSCL domain. The proposed approach consists of identifying, studying and formalizing "Collaborative Learning Patterns": detailed descriptions of well-accepted types of collaborative learning activities defined by Collaborative Learning experts that can eventually be used by software developers to identify software components applicable to several types of component-based CSCL applications. Collaborative Learning Patterns can be understood as a trade-off between the mentioned pure top-down and bottom-up approached for making Collaborative Learning concepts understandable by developers of software applications. The paper is structured as follows: section 2 motivates, defines, and illustrates with two examples the concept of Collaborative Learning Pattern. Section 3 shows how Collaborative Learning Patterns should be used by software developers and describes the authors' experience developing a specific CSCL application. Finally, section 4 summarizes the main conclusions derived from the work presented in this paper and enumerates some important future research issues.

2. Collaborative Learning Patterns

2.1. Motivation

Traditional efforts, shown in Figure 1, for establishing a common ground among experts in the Collaborative Learning domain and software developers include both top-down and bottom-up approaches.

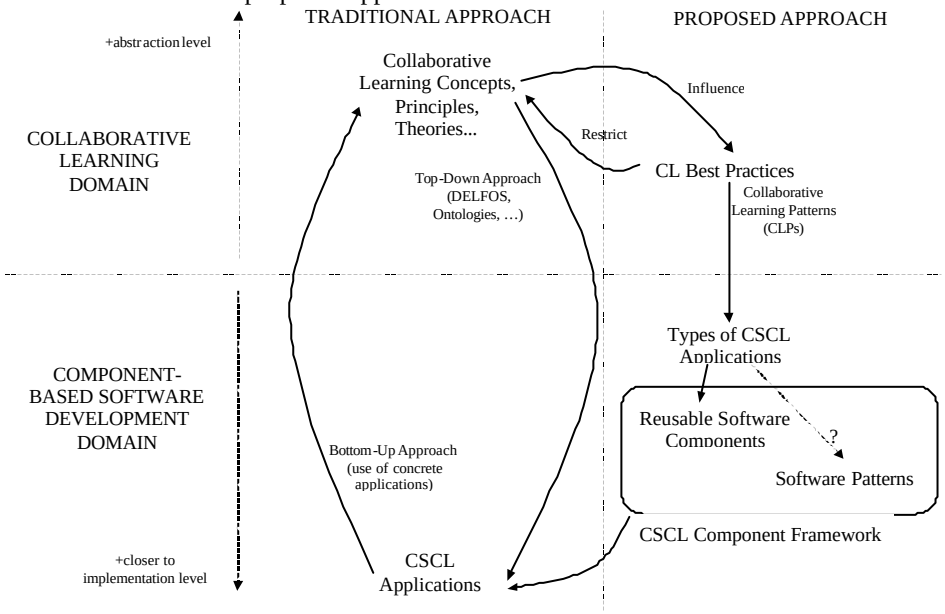


Figure 1. Collaborative Learning Patterns as an alternative for establishing a common ground between the Collaborative Learning domain and the software development field

Some of the most representative approaches in the top-down category are CSCL conceptual frameworks and ontologies.

DELFOS (a Description of a tele-Educational Layered Framework Oriented to Learning Situations) is a CSCL conceptual framework developed by the authors [16]. DELFOS was defined in order to support the complex and interdisciplinary development of applications in the CSCL domain. It proposed a learning model, a generic architecture for CSCL applications, and a development method based on the principles of the participatory analysis and iterative design approaches. The definition of these elements was performed from a holistic perspective, which was very helpful for including important aspects of the learning situations. However, it resulted to be a very complex proposal for its use as a software development methodology. Furthermore, it provided limited help in terms of software reusability, flexibility and customization.

Collaborative Learning Ontologies [3] try to offer a formal shared conceptualization of the domain based on specific theories. Current proposals only include incomplete views of the domain and they do not provide ways of applying the ontological definitions to the support of development efforts of CSCL applications in a practical way.

On the other hand, bottom-up approaches are based on the development of specific CSCL applications that aim at extracting significant elements of the framework. However, the authors experience with this approach [9] shows that identification of reusable components is extremely difficult as the developed applications are biased towards a specific learning problem. Also, it easily becomes evident that a general and reusable formalization is necessary at the domain level. Both facts confirm the problems encountered in the field of Component-Based Software Engineering [5].

Therefore, a new approach, also shown in Figure 1, has been explored based on the use of Collaborative Learning Patterns that will be described in the following subsection.

2.2. Definition

The term "Collaborative Learning Pattern" is derived from the notion of "Collaboration Design Pattern" introduced in [7] and defined as a way of describing "*[…] best practices in collaborative learning*" used as *"[…] a shorthand to effectively communicate collaborative activities, and provide building blocks for more complex situations*" in the CSCL field. The authors of [7] conclude that their patterns offer *"[…] real world examples that can guide technical discussions (some times giving birth to a software structure of the same name)*" but they do not provide clues about how this process could be possible.

Our idea of "Collaborative Learning Patterns" (CLPs) goes a step further in this sense. They can be understood as a way of describing types of collaborative learning activities easily understandable by software developers. CLPs are identified and formalized by Collaborative Learning practitioners (mainly teachers) as well as validated by pedagogy experts. They are intended to be used by software developers in order to derive common requirements for CSCL applications supporting collaborative learning activities of the same type (i.e. activities compliant with the same CLP). In spite of this final use of the CLPs, it is important to point out that the contents of the CLPs themselves do not include any technical information: the types of collaboration activities they describe could be realized without the support of CSCL applications.

CLPs are represented according to a formalism, shown in Table 1, that enlarges the one previously described for "Collaboration Design Patterns" [7]. That table also shows two examples of CLPs, drawn form a larger set that resulted from our analysis, defining well-known practices in Collaborative Learning: jigsaw and pyramid [13].

Table 1. Collaborative Learning Pattern structure and its application to Jigsaw and Pyramid-like activities

Facet	**Explanation**	**Example #1**	**Example #2**
Name	Name of the CLP	Jigsaw	Pyramid
Problem	Learning problem to be solved by the CLP	Complex problem whose resolution implies the handling and/or collection of information that can be easily divided into disjoint sets and that can be used for the resolution of independent subproblems	Complex problem, usually without a specific solution, whose resolution implies the achievement of gradual consensus among all the participants
Example	A real-world learning activity suitable of being structured according to the CLP	Collaborative design of a computing system where the study of each subsystem is assigned to a particular participant	Collaborative proposal of the design of a computing system where each participant contributes with a complete design that is subsequently compared with other contributions and consequently refined
Context	Environment type in which the CLP could be applied	Several small groups facing the study of a large amount of information for the resolution of the same problem	Several participants facing the collaborative resolution of the same problem
Solution	Description of the proposal by the CLP for solving the problem	Each participant in a group (jigsaw group) studies a particular subproblem. The participants of different groups that study the same problem meet in an "Expert Group" for	Each individual participant studies the problem and proposes a solution. Groups (usually pairs) of participants compare and discuss their proposals and, finally, propose a new share solution. Those groups join in

		exchanging ideas. At last, jigsaw group participants meet to solve the whole problem. Each participant contributes with its "expertise"	larger groups in order to generate new agreed proposal. At the end, all the participants must propose a final and agreed solution
Actors	Actors involved in the Collaborative Learning activity described by the CLP	• Teacher • Learner • Evaluator	• Teacher • Learner • Evaluator
Types of Tasks	Types of tasks, together with their sequence, performed by the actors involved in the activity. (NOTE: due to space restrictions only types of tasks performed by learner and teacher are shown)	**Learner:** 1. Access to the information related with the subproblem 2. Individual study of the subproblem 3. Subproblem discussion in the experts group 4. Problem resolution in the jigsaw group 5. Result proposition 6. Process self-evaluation **Teacher:** 1. Global problem definition 2. Division of the problem in subproblems 3. Creation of jigsaw groups 4. Assignment of subproblems 5. Provision of useful information 6. Floor control system establishment 7. Decisions about control of time 8. Activity progress monitoring 9. Result evaluation	**Learner:** 1. Access to the information related with the problem 2. Individual study of the problem 3. Individual solution proposal [REPEAT 4. Formation of groups 5. Group discussion 6. Common solution proposal] (until only one group remains) 7. Process self-evaluation **Teacher:** 1. Global problem definition 2. Provision of useful information 3. Group dimensioning 4. Decisions about control of time 5. Activity progress monitoring 6. Result evaluation
Types and structure of Information	Description of the types of information identified in the collaborative activity and how they are related	• Input information needed for global problem resolution • Partial information assigned to subproblems • Subproblem resolution proposal • Global problem resolution proposal • Correct global problem resolution (optional)	• Input information needed for global problem resolution • Intermediate resolution proposals • Global problem resolution proposal • Correct global problem resolution (optional)
Types and structure of Groups	Description of the types of groups of learners identified in the collaborative activity and how they are related	• Jigsaw groups • Experts groups in charge of subproblems	• Growing pyramid groups

3. From Collaborative Learning Patterns to Software Design Patterns

The nature of the information provided by the definition of CLPs, as shown in the previous section, is suitable for being used by software developers. The information a CLP provides could be used as a source for the derivation of common functional requirements for all CSCL applications devoted to the support of collaborative learning activities of the type defined by the CLP. Obviously, the use of CLPs would depend on the specific software development methodology that is employed. As a way of illustrating these ideas, if a software development methodology based on the Unified Process (UP) [1] is chosen, the information provided by CLPs might be used as the basis for the derivation of actors and use cases, the conceptual model (also known as domain model), and the analysis of the use cases during the iterations of the so-called "Inception Phase". UP has been chosen for the illustration of the usage of CLPs because it is a very common methodology for the development of component-based software applications. Nevertheless it is important to point out that UP is not the only choice for CLPs.

Figure 2 shows UML (Unified Modeling Language) use case diagrams and class diagram representing use cases and the conceptual modeling for a software application that could eventually support a collaborative learning activity of the type described by the jigsaw CLP defined in Table 1. As it can be appreciated, the use case diagram focuses its scope in the reflection of functionality needed for supporting the tasks performed by the different actors involved in the CLP. Although it is not shown here, these use cases have an associated detailed description that must be agreed with the CLP writers in order to check that there is a common understanding of the details and implications of the functional requirements. On the other hand, the conceptual or domain model reflects the types and the structure of the information and groups described by the CLP, as well as the interrelation among them. It can be appreciated, for instance, how *Jigsaw Group*, and *Expert Group* classes are associated to *Global Problem* and *Subproblem* classes which, at the same time, maintain an aggregation association between them.

After completing the UP inception phase using the information provided by CLPs, and using normal software development techniques prescribed, in this example, by UP, it is possible to obtain a software design architecture

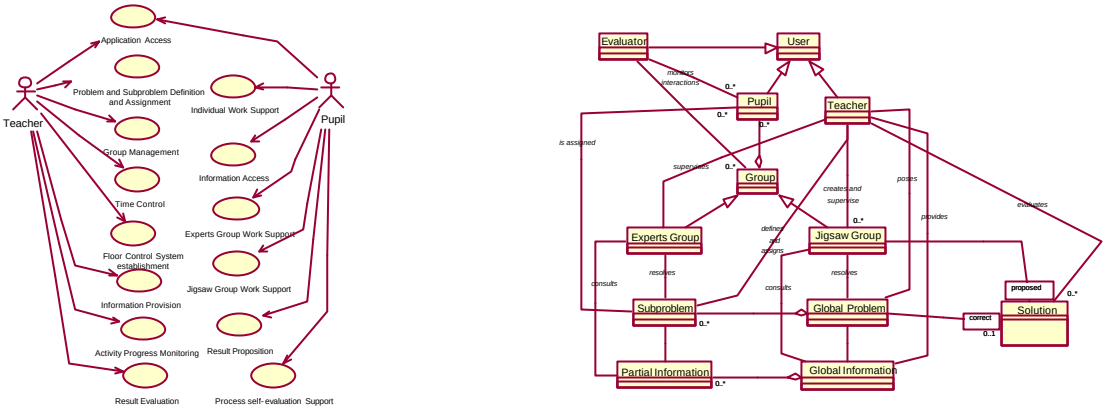
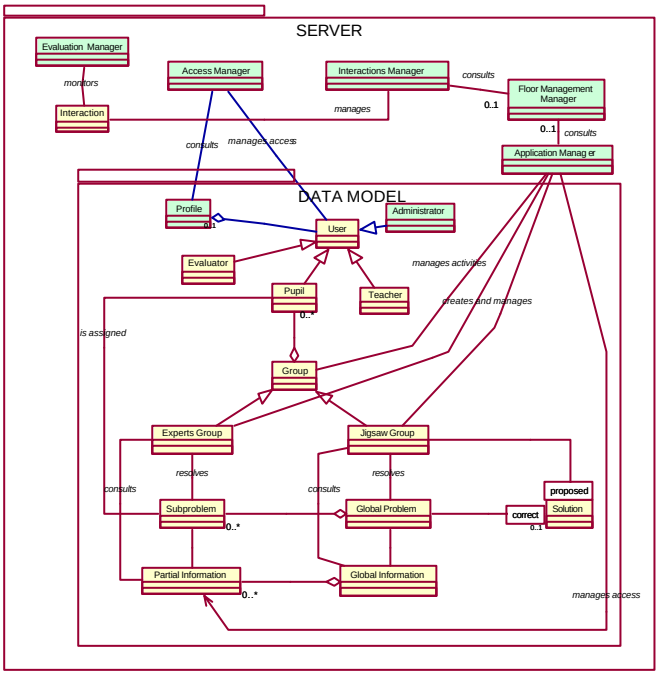


Figure 2 UML use case diagram and conceptual class diagram derived from the *jigsaw* CLP

for a jigsaw-like CSCL application. Figure 3 shows the static view of such a potential architecture.

In the software design architecture shown in Figure 3 it is possible to identify several candidate software components such as: Access Manager, Floor Control Manager, Interaction Manager and so on. Of course, it is difficult to prescribe a way of starting with a CLP definition and reaching a specific software design for the corresponding type of applications. In other words, different valid designs can be obtained from the same CLP. Nevertheless, our experience when developing CSCL applications indicates that CLPs are a very useful tool during the first stages of the development. Furthermore, software designs that we have obtained have successfully been reused in more than one application. This indicates that, by starting with Collaborative Learning Patterns, it could eventually be possible to obtain valid Architectural Software Patterns (of course, once a proper number of applications based on the same CLP have been developed and validated). This is still an open research issue.



In terms of software reusability, the implications from the presented approach are very important: the use of CLPs help software developers to understand the requirements and involved concepts of specific types of CSCL

Figure 3. UML class diagram showing the static view of a potential architecture of a CSCL application that supports a jigsaw-like collaborative learning activity

applications. Therefore, it is much easier to identify common software components for those types of application. These common components are potentially more reusable that those obtained when developing a particular CSCL application not bound to a CLP. This fact facilitates the progressive fulfillment of the original goal of obtaining a component framework for the CSCL domain.

In terms of usability and meaningfulness from the point of view of Cognitive and Learning Sciences experts, the CSCL applications developed by starting from CLPs reflect solid Collaborative Learning principles, while they also comply with best practices widely understood by education practitioners. Although CLPs have a very limited scope when compared with the great amount of concepts and theories that belong to the Collaborative Learning field, CLPs and their proposed use by software engineers provide a realistic path to the use of a subset of concepts of certain importance. The approach described in this paper is based on the experience of our group in the development of CSCL applications during the last decade. It has been applied, for example, to the development of a component-based CSCL application devoted to the support of a course on computer design for Telecommunications Engineers in our University. That application, called eLAO, supports several collaborative learning activities that belong to both the jigsaw and the pyramid CLPs. In this case, we have been able to use the proposed approach to a fusion of two different CLPs, showing that reusability is not necessarily reduced to applications belonging to the same CLP. Reusability of the software components developed for eLAO is currently being evaluated in the construction of other CSCL applications based on the same CLPs. Preliminary conclusions indicate that components that support the teacher's tasks (e.g., student and group management, task assignment,…) and those components related to interaction and information handling (e.g., floor management, interaction management, …) are the most reusable in applications based on a same CLP.

4. Conclusions and Future Work

This paper has introduced and illustrated the concept of Collaborative Learning Pattern (CLP) as a promising approach for establishing a common ground among experts and practitioners from Cognitive and Learning Sciences and software developers of CSCL applications. CLPs can be used by software developers during the first stages of software development methodologies in order to understand common functional requirements of different types of CSCL applications. In subsequent stages, they can also be used for the identification of common software components for CSCL applications that support collaborative learning activities compliant with a particular or a combination of existing CLPs. These identified components will eventually belong to a general CSCL component framework for facilitating reusability, flexibility and customization of CSCL software. The paper has also presented two examples of CLP definition and an example of how those particular CLPs were used by software developers in order to identify software components applicable to a particular component-based CSCL application. Another CLP (simulation) has also been defined (although not presented here due to space restrictions) and the combination of two CLPs has been successfully employed during the development of a specific CSCL application.

This paper has presented an open research effort that still has to face several challenges. Currently, the improvement in software component reusability obtained by the CLP approach is being evaluated. Also, the CLP definition formalism is being discussed with learning experts in order to include more useful information for software developers. An evaluation of the CLP approach from the viewpoint of traditional knowledge engineering techniques such as CommonKADS is also in progress. At the same time, new CLPs are being defined in order to find potential limitations of the approach. A very interesting research issue under study is the identification of ways of achieving an automatic or semiautomatic translation of CLPs into software development artifacts. In other words, we are currently trying to propose the conditions and the steps of a methodology that would allow to derive Software Design Patterns (or the selection of existing ones) from Collaborative Leaming Patterns. Another possible improvement of the CLP approach deals with the introduction of specific information about the types of interactions to register and analyze in order to support coaching and evaluation aspects, of major importance in the CSCL field [12].

Acknowledgements

The authors want to acknowledge the contributions from other members of the EMIC Group (Education, Media, Information, and Culture), specially J.L. Barrio, B. Rubia, D. Hernández, P. Orozco, and R. Anguita. This work was partially financed by the Autonomous Govemment of Castilla and León, Spain (project VA117/01), and the Ministry of Science and Technology, Spain (projects TIC2000-1054 and TIC-2002-04258-C3-02).

References

[1] Arlow, J. and Neustadt, I. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison Wesley Professional, 2001.

[2] Askit, M., Marcelloni, F., and Tekinerdogan, B., Developing Object-Oriented Frameworks Using Domain Models *ACM Computing Surveys*, vol. 32, 2000.

[3] Barros, B., Verdejo, M. F., Read, T., and Mizoguchi, R., "Applications of a Collaborative Learning Ontology," *Proceedings of the Mexican International Conference on Artificial Intelligence (MICAI'02)*, 2002.

[4] Brugali, D. and Sycara, K., Frameworks and Pattern Languages: an Intriguing Relationship *ACM Computing Surveys*, vol. 42, 2000.

[5] Carey, J. and Carlson, B., Lessons learned becoming a framework developer *Software Practice and Experience*, vol. 43, pp. 789-800, 2002.

[6] Crnkovic, I., Hnich, B., Jonsson, T., and Kiziltan, Z., Specification, Implementation, and Deployment of Components *Communications of the ACM*, vol. 45, pp. 35-40, Oct, 2002.

[7] DiGiano, C., Yarnall, L., Patton, C., Roschelle, J., Tatar, D., and Manley, M., "Collaboration design patterns: conceptual tools for planning for the wireless classroom," *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'02)*, 2002.

[8] Dillenbourg, P. *Collaborative Learning: Cognitive and Computational Approaches*, Oxford, UK: Elsevier Science, 1999.

[9] Dimitriadis, Y. A., Asensio, J. I., Toquero, J., Estébanez, L., Martín, T. A., and Martínez, A., "Towards a Software Components System for the Computer-Supported Collaborative Learning Domain (in spanish)," *Proceedings of the Spanish Informatics and Telecommunications Conference (SIT'02)*, Seville, Spain, 2002.

[10] Fach, P. W., Design Reuse through Frameworks and Patterns *IEEE Software*, pp. 71-76, Sep, 2001.

[11] Hopkins, J., Component Primer *Communications of the ACM*, vol. 43, pp. 27-30, Oct, 2000.

[12] Jermann, P., Soller, A., and Muehlenbrock, M., "From Mirroring to Guiding: A Review of the State of the Art Technology for Supporting Collaborative Learning," *Proceedings of ECSCL 2001*, 2001.

[13] Johnson, D. W. and Johnson, R. T. *Learning together and alone: cooperative, competitive and individualistic learning*, Allyn and Bacon, 1999.

[14] Koschmann, T. Paradigm shift and instructional technology. In: *CSCL: Theory and Practice of an emerging paradigm*, ed. Koshmann, T. Lawrence Erlbaum, 1996. pp. 1-23.

[15] Mili, H., Fayad, M., Brugali, D., Hamu, D., and Dori, D., Enterprise frameworks: issues and research directions *Software Practice and Experience*, vol. 32, pp. 801-831, 2002.

[16] Osuna, C. and Dimitriadis, Y., "A framework for the development of educational collaborative applications based on social constructivism," *Proceedings of the CYTED RITOS International Workshop on Groupware (CRIWG'99)*, 1999.

[17] Roschelle, J., DiGiano, C., Koutlis, M., Repenning, A., Phillips, J., Jackiw, N., and Suthers, D., Developing Educational Software Components *Computer*, pp. 50-58, Sep, 1999.

[18] Roschelle, J., Kaput, J., Stroup, W., and Kahn, T. M., Scalable integration of educational software: exploring the promise of component architectures *Journal of Interactive Media in Education*, vol. 98, Oct, 1998.