Position Statement: From Collaborative Learning Patterns to component-based CSCL Applications

Juan I. Asensio[1], Yannis A. Dimitriadis[1], Marta Heredia[1], Alejandra Martínez[2], Francisco J. Álvarez[1], María T. Blasco[3], César A. Osuna[4]

[1]Department of Signal Theory, Communications and Telematics Engineering
{juaase,yannis,mherrod,fraalv}@pireo.tel.uva.es
[2]Department of Computer Science
amartine@infor.uva.es
[3]Faculty of Education
tbq@dlyl.uva.es

University of Valladolid
Camino del Cementerio s/n
47011 Valladolid. Spain

[4]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 Cognitive Educational Sciences experts and those used by software developers. In order to alleviate this problem, this position statement 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 and Motivation

Our multidisciplinary research group at University of Valladolid, Spain, is formed by people coming from the Faculty of Education and the Schools of Informatics and Telecommunications Engineering. Our main research interest is focused on the field of Computer-Supported Collaborative Learning (CSCL).

Some of our currently open research lines, within CSCL, include the computational support of evaluation tasks in collaborative learning scenarios, the suitability of grid computing infrastructures for CSCL applications and, more closely related to the topic of this statement, the application of Component-Based Software Engineering (CBSE) principles to the development of flexible, customizable and reusable CSCL software.

With respect to the combination of CBSE and CSCL, one of the most important problems we are facing 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]. 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.

In this sense, traditional efforts for establishing a common ground among experts in the Collaborative Learning domain and software developers include both top-down and bottom-up approaches. 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 [10]. DELFOS was defined in order to support the complex and interdisciplinary development of applications in the CSCL domain. It proposed a learning model, some ideas about a generic architecture for CSCL applications, and a development method based on ideas from the participative analysis and design. Nevertheless, DELFOS was mainly focused on the

learning model. The complexity and broad scope of the leaming model made the software development methodological proposals very little usable. Furthermore, in terms of software reusability, flexibility and customization it provided limited help.

Collaborative Learning Ontologies [3] try to offer a formal shared conceptualization of the domain based on concrete 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 concrete CSCL applications that aim to extract significant software components. However, the authors experience [6] with this approach shows that identification of reusable components is extremely difficult as the developed applications are very influenced and biased towards a concrete 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 [4].

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 section 2. Two concrete examples of Collaborative Leaming Patterns will also be presented in that section. Section 3 contains our position statement about the usefulness and possible limitations of the Collaborative Learning Pattern approach for the development of CSCL applications. Finally, section 4 concludes this document and describes some future research items.

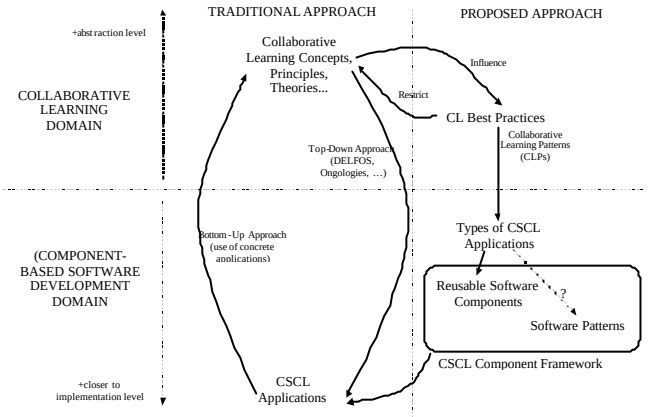

Fig. 1. Collaborative Learning Patterns as an alternative for establishing a common ground between the Collaborative Learning domain and the software development field

2 Collaborative Learning Patterns: Definition and Examples

Our proposition of the term "Collaborative Learning Pattern" is derived from the notion of "Collaboration Design Pattern" introduced in [5] 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 [5] 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 don't 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.

We represent CLPs are according to a formalism, shown in Table 1, that enlarges the one previously described for "Collaboration Design Patterns" [5]. That table also shows two examples of CLPs, drawn form a larger set that resulted from our analysis, defining well-known practices in Collaborative Leaming: jigsaw and pyramid [9].

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 concrete 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 lot 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 exchanging ideas. At last, jigsaw group participants meet to solve the whole problem. Each participant contributes with its "expertise"	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 shared solution. Those groups join in 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 • Pupil • Evaluator	• Teacher • Pupil • Evaluator
Types of Tasks	Types of tasks, together with their sequence, performed by the actors involved in the activity.	**Pupil:** 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	**Pupil:** 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 pupils identified in the collaborative activity and how they are related	• Jigsaw groups • Experts groups in charge of subproblems	• Growing pyramid groups

3 Position Statement: From Collaborative Learning Patterns to component-based CSCL applications

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 concrete 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 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 appreci-

ated, 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.

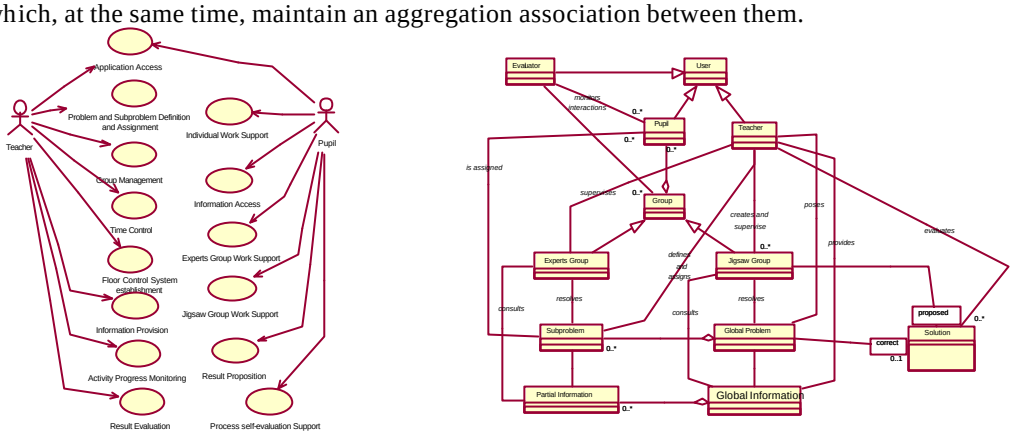

Fig. 2. UML use case diagram and conceptual class diagram derived from the jigsaw CLP

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 for a jigsaw-like CSCL application

Of course, it is difficult to prescribe a way of starting with a CLP definition and reaching a concrete software design for the corresponding type of applications. In other words, the final design of the CSCL application largely depends on the abilities of the involved software developers. Nevertheless, our experience 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 and that indicates that, by starting with Collaborative Learning Patterns, it could eventually be possible to obtain valid Architectural Software Patterns [7]. Nevertheless, it is premature to state that CLPs always result in valid Software Patterns: 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 helps software developers to understand the requirements and involved concepts of concrete types of CSCL 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 significance from the point of view of Cognitive and Learning sciences experts, the CSCL applications developed by starting from CLPs strongly reflect solid Collaborative Learning principles while they also reflect 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 the proposed use by software engineers provide a realistic path to the use of a subset of certain importance.

The approach described in this document is based on the experience of our group in CSCL appications during the last decade. It has been applied, for example, to the development of a concrete 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 necessarilly 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 and those components related to information handling are the most reusable in applications based on the same CLP.

4 Conclusions and Future Work

This document 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. In our opinion, 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 CLP. These identified components will eventually belong to a general CSCL component framework for facilitating reusability, flexibility and customization of CSCL software. This document 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 open research effort still has to face several challenges. Currently, the software component reusability improvement obtained by the CLP approach is being evaluated. Also, the CLP definition formalism is being discussed with learning experts in order to include more information useful for software developers. 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 from Collaborative Learning 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 [8].

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 Government 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 Proffesional (2001)
2. Askit, M., Marcelloni, F., and Tekinerdogan, B.: Developing Object-Oriented Frameworks Using Domain Models. ACM Computing Surveys 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. Carey, J. and Carlson, B.: Lessons learned becoming a framework developer. Software Practice and Experience 43 (2002) 789-800
5. 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)
6. Dimitriadis, Y. A., Asensio, J. I., Toquero, J., Estébanez, L., Martín, T. A., and Martínes, 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)
7. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
8. Jermann, P., Soller, A., and Muehlenbrock, M.: From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. Proceedings of ECSCL 2001. (2001)
9. Johnson, D. W. and Johnson, R. T.: Learning together and alone: cooperative, competitive and individualistic learning. Allyn and Bacon, Needham Heights, MA (1999)
10. 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. (1999)