IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4603

UML Modeling for Cooperative Problem-Based Learning Situations: Towards Educational Components

Pierre Laforcade, Franck Barbier
Computer Science Research Institute of University of Pau et des Pays de l'Adour , LIUPPA, Université de Pau, BP 1155, 64013 Pau CEDEX FRANCE, Phone: (+33)5.59.92.33.43, Fax: (+33)5.59.80.83.74, E-mail: pierre.laforcade@univ-pau.fr

ABSTRACT

Problem-Based Learning Situations require accurate template models in which the roles of tutor and learner participate in varied codified cooperative activities. This paper discusses the use of the UML to first build such customizable models, and next to derive Educational Software Components from models. The paper contributes to reduce the lack of flexibility in open distance learning tools where distribution of components applies with some difficulty. It is on purpose introduced the designer role for problem-based learning situations. This designer aims to assemble educational components in order to offer computer-aided learning supports. Model examples and techniques to implement components are also briefly evoked.

Keywords: *Educational Component, Cooperative Learning, Problem-Based Learning Situation, UML.*

1 INTRODUCTION

Our work copes with learning situations involving cooperation between tutors and learners. Within this context, computer-aided learning hinges on software and platforms whose customization allows to implement scenarios embodying such cooperation. Because of the monolithic aspect of learning platforms and software, as well as the specificity of education based on Problem-Based Learning Situations (PBLS)(Meirieu, 1988), we propose in this paper an approach using the notion of Educational Software Component (Roschelle et al., 1999). PBLS rely on cognitive models of cooperative activities for tutors and learners. These cognitive models can be specified once for all and captured within components. By offering enough flexibility, namely parameterization to keep a good degree of tuning, and by naturally supporting distribution, Educational Components, when reused, allow to assemble new PBLS in software systems. Distance learning issues via distribution are especially associated with the idea of software component.

We focus in this paper on the UML specification of Educational Components. This formalism favors, at a conceptual level, the description of tutor/learner and learner/learner cooperation. At implementation time, UML supplies Component & Deployment Diagrams to package and deploy specification pieces into components. We sketch in this paper such cooperation and briefly discuss at the end of the paper, implementation based on a dedicated library. Indeed, we divide the modeling of pedagogical activities into Statechart Diagrams and therefore illustrate how to easily and quickly implement these dynamical models in Java.

2 EDUCATIONAL ENGINEERING

Educational Engineering covers techniques and tools that assist, and possibly automate in software, the universal and dual actions of teaching and learning. In this section, after describing our context of work and goals, we walk through current innovative projects in this domain.

2.1 Context of Work

Educational Components here described appear within the framework of a more general project: the specification of an environment allowing a teacher to implement cooperative learning situations. Our first goal is to help teachers to specify learning situations. PBLS are quite different from classic learning approach based on the notions of courses, exercises, assessments. PBLS are indeed based on the idea of cooperative activity and more exactly cooperative resolution of problems.

Recent works (Nodenot et al., 2002) describe stakes, actors and application principles of such learning activities. We pay attention on the definition of a system (called *Learning Management System* or *LMS*) that manages activities for distant user communities (learners and tutors). We also show that a pedagogy relying on cooperative activities conforms to normalization works carried out by international e-Learning consortia (AICC, 2000; ARIADNE, 2001; Dublin_Core, 2000; GESTALT, 2000; IMS, 2000; ISO/IEC_JTC1_SC36/WG2, 2001; LTSC, 2000; MASIE_Center, 2002; PROMETEUS, 2000; SCORM, 2000).

Previously, we worked on the specification of a role-component library enabling a designer to reuse learning scenarii (Sallaberry et al., 2002). This approach allows the assembling of new roles by combining pre-existent role-components.

2.2 Educational Components

Over years, research in Educational Engineering emphasizes the support of interoperable and reusable applications as well as "electronic" services (Wiley, 2000). The Educational Component (EC) approach is growing: "*having component developers collaborate with domain experts to build applications may be the future of software development*" (Roschelle et al., 1999).

The component paradigm used within the Educational domain has numerous objectives:

1. sharing learning resources between software tools and systems,
2. making interoperability between tools,
3. making interoperability between applications and learning resources,
4. reusing learning resources (by teachers),
5. reusing educational software (by developers).

As the word "Component" in Software Engineering, "Educational Component" has multiple meanings and may have very different interpretations. Thus, we cannot find out a generic definition of an EC. We point out the dual notion of Learning Object (LO). LOs are "*any digital resource that can be reused to support learning*" (Wiley, 2000) or "*Learning Objects are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning*" (LOM, 2000). In fact, such a fine-grained component supports learning by means of its embedded learning content. So, research works on LOs rather correspond to points 1-3-4 above (ARIADNE, 2001; De_La_Passardière et al., 2001; IMS, 2000; Koper, 2001; LOM, 2000; SCORM, 2000) and relate to classical learning concepts as courses, exercises, assessments. LO approach is more convenient for the common learning approach than for the PBLS constructivist one (Deschênes et al., 1996) – learner build his own knowledge by doing.

2.1 Current Trends and Directions for Educational Software Components

We study different projects based on Educational Software Components that we split into two categories.

ESCOT (Educational Software Components Of Tomorrow) project (ESCOT, 2002) aims at the construction of a digital library containing educational software relating to Middle School mathematics. One of the ESCOT's goals is to have interactive JavaBeans-based content within an educational context. In the same way, ESCOT explores the process of distributed software-development with the specific objective to rapidly build and deploy reliable software (Repenning et al., 2001). It also deals with the EC stemming from AgentSheets (Agent_Sheets, 2001) and E-Slate projects (Birbilis et al., 2000). AgentSheets is an authoring-tool for the creation of reusable EC under the shape of applets, directly integrable on a Web page. These components can be used for exercises of simulation, demonstration, scientific modeling, etc. The technology used is simple and the integration on a Web page implies that the component is only a pedagogical element of a bigger one. In E-Slate, components are supplied in the form of prefabricated objects (card, clock, vector…). They possess a mechanism of interconnection (glue) and are configurable, customizable: they can contain features for a specific domain. They are connected together within the same global application which allows to visually assemble them according to "the puzzle's analogy"; they especially have an appropriate graphical interface.

The works on SimulNet (Anido et al., 2001) propose a layered component model for the support of Web-based interactive and collaborative applications (framework) as well as a development of an educational application based on this model. Every component aims at supplying a feature required in Web-based collaborative applications. In a similar way, the global objective of the PLACE project (PLAteforme à Composants Evolutive) (Peter et al., 2002) is the following study: how to realize flexible cooperative working environments based on models of standard components. In both SimulNet and PLACE projects, the Software Component principles are applied in the design and the realization phases: the creation of a Web environment dedicated to distant learning. For the PLACE project, the architecture is based on the use of the EJB / J2EE platform in order to build a CSCW (Computer Supported Cooperative Work) platform whereas SimulNet proposes a client / server architecture. At the structure level, the analogy between component and tool (auditing tool, e-mail, bulletin board, chat, whiteboard, agenda, project management, event delivering, producer-consumer manager…) is present in both projects.

The various researches previously quoted have in common a component approach based upon the Software Engineering. So, an Educational Component is close to the notion of tool. Like a Software Component, an Educational Component supplies services and has to be assembled with other ones in order to build an educational application.

3 EDUCATIONAL COMPONENT MODELING

This part describes first of all our viewpoint on EC, according to our project's requirements. Then, we detail how we may represent a pedagogical activity. This activity will be grounded on our EC notion. Then, we provide an illustrated approach in order to highlight EC main characteristics within a PBLS and then we show the building steps of such a component. Finally, we sketch an EC implementation example based on a Java library.

3.1 Properties of Educational Components

Like the PLACE and SIMULNET projects, our vision of Educational Component is close to a component / tool: a black box supplying services. However, our EC approach does not concern the architecture of the PBLS system. In contrast, it allows an EC manipulation (composition or extension) in order to *design new learning activities*. So, it will be possible to build opened and flexible cooperative PBLS, easily modifiable by the end-users: teachers / designers. Our EC concept does not embed educational contents like LOs but offers "pedagogical services": resources exchange service, synchronous communication service between tutor/learner, etc.

EC describes a small pedagogical activity process. It corresponds to a generic and reusable PBLS element:

- It supplies *pedagogical services* (monitoring, regulation, production…services).
- It describes *one or several views:* the learner's view, the tutor's view, … (see following example).
- It is *configurable* during its use / assembly: some *generic* parameters have to be instantiated, insuring a better integration in the pedagogical activity. This configuration allow the EC to match specific learning situation.
- It is *customizable:* for example, the designer may choose between synchronous or asynchronous characteristic of a conversational EC.

On one hand, our EC model gives pedagogical services required by teachers. On the other hand, our model is supported by tool features: they can be supplied by any Software Components (chat, e-mail, diary…).

3.2 Modeling Method

Information System modeling based on UML is a central topic in recent papers. In (Nodenot et al., 2002; Sallaberry et al., 2002) we used UML to describe cooperative PBLS in terms of actions, roles, ressources, learning objectives and pedagogical activities. In this paper, we focus on the details of the pedagogical activity. To that extent, like (Bourguin, 2000), we refer on the Activity Theory (Engeström et al., 1998).

Our original contribution is our choice to represent pedagogical activities (learning or tutoring) with UML Statecharts. Such a Statechart represents the *expected progresses* for a user (learner and tutor) within the pedagogical activity. Here is the analogy between statechart elements and pedagogical activities:

- Any *state* of a diagram represents a *step* in the expected progresses of an activity.
- A *transition* represents a *pedagogical action* that a user can perform in its activity. The transitions also allow a *non-linear meshing* of progress possibilities within the activity. They are composed of:
- An *event*: it is raised by the user or the system in order to *validate an expected action*
- A *guard*: it represents a required condition whether the associated event is raised.
- *Action*: it is a call to a service provided by a tool – users have tools at their disposal.
- Generalization and aggregation of states allow to reliably divide a pedagogical activity into sub-elements. This insures a structural and hierarchical design.

During the execution stage, statechart models enable the LMS (*Learning Management System*) to provide users with context-sensitive tools: at each state of the users activity, the system will know which service calls are enabled / disabled by the designer. Consequently, the system will be able to enable / disable tools functionalities.

Statechart modeling also allows to manage a history (trace) of the various states, transitions, events, etc, that the users passed through. So, during the PBLS execution stage, it will be possible for both system and tutors to supervise *the actual pedagogical activity*. This allows to improve the tutoring and can be used in order to build a learner profile. This profile is used to regulate the pedagogical activity. It is also used in order to re-route a user towards another activity[1].

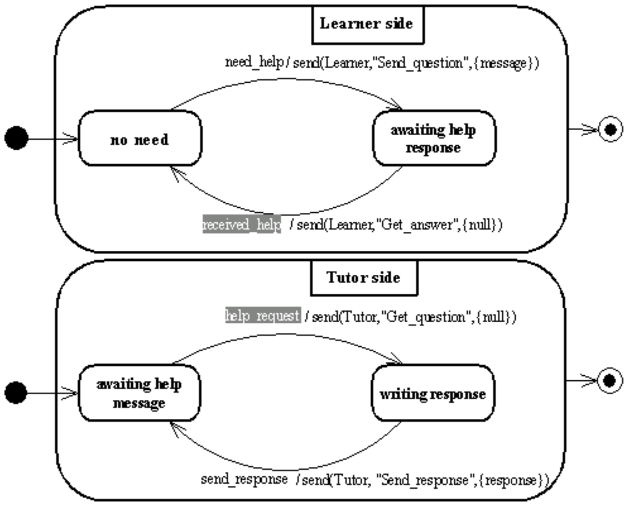
3.3 Illustration

The objective here is to highlight and to formally describe low level pedagogical activities. These activities will contribute to describe high level pedagogical activities. Thus, any pedagogical activity results from an assembly process of sub-activities.

In order to identify these reusable pedagogical activities, we are inspired of the "four-leaved clove" (David, 2001). This model presents four embedded spaces for the classification of cooperative work activities – production, communication, conversation and coordination. For example, some pedagogical activities which would inevitably appear in cooperative PBLS are: asynchronous or synchronous conversation between learner / learner or learner / tutor, collective production between learners, information pooling, information sharing, information research in a library / Web...

We describe in this part a simple example of our EC model. This component contains a monitoring pedagogical activity called "help on inquiry". This activity consists in giving the possibility for a learner to ask for some help to a distant tutor. This last one can then answer her/him.

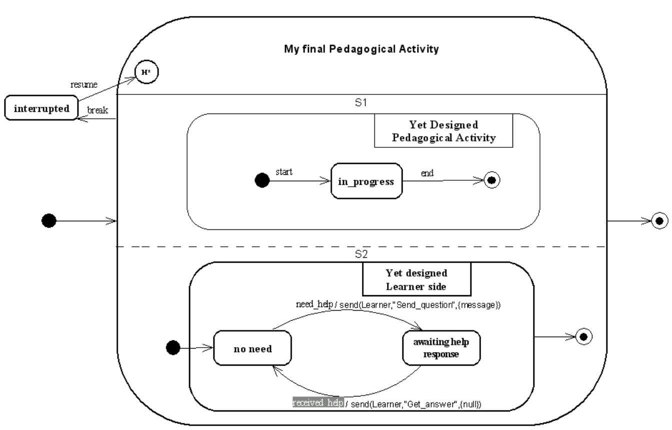
Figure 1 : Example of pedagogical Statecharts into our basic Educational Component



This pedagogical activity concerns two actors: a learner and a tutor. Thus, the pedagogical activity representation requires two Statecharts for each possible pedagogical treatments linked to one of the actors (Figure 1). The required precision for this component configuration is: who are the tutor and learner? The answer will enable the instanciation of the generic parameters "*Tutor*" and "*Learner*". During the design phase, the configuration may also concern the types of the messages: synchronous (*instant messaging*) or asynchronous *(e-mail)*. Moreover, this figure shows two different kinds of events: "*need help*" that corresponds to an event generated by the user "*Learner*"; "*received help*" (grayish event) that corresponds to an independent event of the "*Learner*", it is generated by the LMS or another user ("*Tutor*"). Every transition involves an action-element of this shape: *send (Target, Method, Parameters) - method* objects correspond to services provided by the *target*.

The following figure shows an example of assembly between the concern learner part ("*Yet designed Learner Side*") and a statechart ("*Yet designed Pedagogical Activity*") representing the pedagogical activity predicted by the PBLS designer. This second statechart is deliberately limited to one and only state ("in_progress") in order to hide pedagogical activity complexity. Both statecharts are assembled by aggregation into a new global state ("*My Final Pedagogical Activity*"). It means that both sub-statecharts can evolve separately during the actual realization of the pedagogical activity: while following sub-statechart activity, the learner may request some help and wait for the

Figure 2 : An assembly example between yet designed pedagogical activity and Statechart from EC



tutor answer. This "concurrent" mode is a pedagogical assembly possibility but other alternatives remain as direct chaining of two statecharts.

3.4 Implementations

We sketch here how our statecharts are implemented in order to become operational components. Here, the statechart of Figure 2 ("*My final Pedagogical Activity*") is build and execute in the same way as a simulation-tool in order to validate our model. To this end, we use the *PauWare* Statechart Java library.

```
protected Tutor _tutor;
…
protected Statechart _interrupted;
protected Statechart _S1;
protected Statechart _S2;
// Statechart_monitor extends Statechart class with new transition features
protected Statechart_monitor _My_final_pedagogical_activity;
…
_My_final_pedagogical_activity = new Statechart_monitor((_S1.and(_S2)).xor(_interrupted));
// S1 & S2 states are concurrent à "and" assembling
// "interrupted" state is in exclusive or with the assembled S1&S2 à "xor" assembling
```

Events influence and conduct the way by which learning activities may run:

```
synchronized public void need_help() {
try {
    // transition "need_help" build between states from "_no_need" to "_awaiting_help_response"
_My_final_pedagogical_activity.fires(_no_need,_awaiting_help_response,true,_tutor,"Send_question", null);
    // simulation of the generated event à transition execution
    _My_final_pedagogical_activity.used_up();
}
catch(StatechartException se) {
System.err.println(se.getMessage());
System.exit(1);
}
}
```

4 CONCLUSION

In this paper, we introduce the idea of Educational Component in order to support computer-aided pedagogical activities. We review some current research works on this special concept of Educational Component. Our approach converges towards the modeling of Educational Components with the UML as well as the potential associated implementation of these components. Capturing learning/teaching activities that are in essence cognitive processes, comes up against inappropriate formalisms. We think that a suitable formalism is together helpful for representing all aspects of PBLS and easily leads to concrete software entities. We thus focus on the segmentation of activities in order to deliver components that can be readily assembled and deployed to provide opened and flexible distance learning activities.

Perspectives of our work rely on the use of component models like CCM (CORBA Component Model), EJB or .NET.

5 REFERENCES

Agent_Sheets. (2001). *Getting Started with AgentSheets*. Retrieved, from the World Wide Web: http://agentsheets.com/Documentation/windows/Getting-Started.pdf

AICC. (2000). *Aviation Industry CBT Committee (AICC). document #CMI001: CMI Guidelines for Interoperability v3.4; released* [WWW Document]. Retrieved, from the World Wide Web: http://www.aicc.org/

Anido, L., Llamas, M., Fernandez, M. J., Caeiro, M., Santos, J., & Rodriguez, J. (2001, may 1-5 2001). *A Component Model for Standardized Web-based Education.* Paper presented at the WWW'10, Hong Kong, ACM 1-58113-348-0/01/0005.

ARIADNE. (2001). *Alliance of Remote Instructional and Distribution Networks for EuropeAlliance of Remote Instructional and Distribution Networks for Europe*. Retrieved, 2001, from the World Wide Web: http://www.ariadne-eu.org/

Birbilis, G., Koutlis, M., Kyrimis, K., Tsironis, G., & Vasiliou, G. (2000). *E-Slate: A software architectural style for end-user programming.* Paper presented at the 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland

Bourguin, G. (2000). *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE.* Thèse de doctorat de l'Université des sciences et technologies de Lille. N°2753.

David, B. (2001). IHM pour les collecticiels. In E. Hermès (Ed.), *Les télé-applications* (Vol. volume 13).

De_La_Passardière, B., & Giroire, H. (2001). XML au service des applications pédagogiques. *Revue Sciences et Techniques Educatives, EIAO'01*, pp 99-112.

Deschênes, A. J., Bilodeau, H., Bourdages, L., Dionne, M., Gagné, P., Lebel, C., & Rada-Donath, A. (1996). Constructivisme et formation à distance. *DistanceS, Vol 1, num 1*.

Dublin_Core. (2000). *The Dublin Core Metadata Initiative*. Retrieved, from the World Wide Web: http://purl.org/dc/

Engeström, Y., Miettinen, R., & Punamäki, R.-L. (1998). Perspectives on activity theory. *(Eds.) Cambridge: Cambridge University Press*.

ESCOT. (2002). Retrieved, from the World Wide Web: http://www.escot.org/overview.html

GESTALT. (2000). *Getting Educational Systems Talking Across Leading Edge Technologies*. Retrieved April 1, 2000, from the World Wide Web: http://www.fdgroup.co.uk/gestalt/about.html

IMS. (2000). *The IMS Project*. Retrieved, from the World Wide Web: http://www.imsproject.org/

ISO/IEC_JTC1_SC36/WG2. (2001). *Toward common understanding for Collaborative Learning - from the viewpoint of technical standardization*. Retrieved, from the World Wide Web: http://jtc1sc36.org/doc/36N0153.pdf

Koper, R. (2001). *Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind EML. First draft, version 2*: Educational Expertise Technology Centre, Open University of the Netherlands.

LOM. (2000). *LOM (Learning Object Metadata) working draft v4.1*. Retrieved, from the World Wide Web: http://ltsc.ieee.org/doc/wg12/LOMv4.1htm

LTSC. (2000). *Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Comittee (LTSC)*. Retrieved, from the World Wide Web: http://ltsc.ieee.org/

MASIE_Center. (2002). *Making Sense of Learning Specifications & Standards: A Decision Maker's Guide to their Adoption.*

Meirieu, P. (1988). Guide méthodologique pour l'élaboration d'une situation-problème. In P. E. e. éd. (Ed.), *annexe to : Apprendre… oui, mais comment ?* : Collection Pédagogies.

Nodenot, T., Marquesuzaà, C., Laforcade, P., Bessagnet, M.-N., & Sallaberry, C. (2002, 13-15 november). *Spécifications d'un environnement Web supportant des activités coopératives d'apprentissage.* Paper presented at the Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, Lyon, pp. 253-262

Peter, Y., Vantroys, T., & Viéville, C. (2002). *PLAteforme à Composants Evolutive (PLACE)*. Retrieved, from the World Wide Web: http://noce.univ-lille1.fr/~ypeter/PLACE/Place.pdf

PROMETEUS. (2000). *Promoting Multimedia access to Education and Training in European Society*. Retrieved, from the World Wide Web: http://prometeus.org/

Repenning, A., Loannidou, A., Payton, M., Ye, W., & Roschelle, J. (2001). Using Components for Rapid Distributed Software-Development. *IEEE Software, march/april*, pp 38-45.

Roschelle, J., DiGiano, C., Koutlis, M., Repenning, A., Phillips, J., Jackiw, N., & Suthers, D. (1999). Developing Educational Software Components. *IEEE Computer, 32*(9), 2-10.

Sallaberry, C., Nodenot, T., Marquesuzaà, C., Bessagnet, M.-N., & Laforcade, P. (2002, 27 - 30 May, 2002.). *Information modelling within a Net-Learning Environment.* Paper presented at the 12th Conference On Information Modelling and Knowledge Based, Krippen, Swiss Saxony, Germany

SCORM. (2000). *The Department of Defense Advanced Distributed Learning — ADL — Initiative Releases Version 1.2 of the Sharable Courseware Object Reference Model — SCORM 1.2 —*. Retrieved, from the World Wide Web: http://www.adlnet.org/Scorm/scorm.cfm

Wiley, D. A. (2000). *Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy* (In D. Wiley ed.): Bloomington: Association for Educational Communications and Technology.

(Footnotes)

[1] A re-routed activity is an individual pedagogical activity dynamically allocated; this allows the system to propose new pedagogical activities every time a user finishes an activity and is waiting for other ones, avoiding passive delay.