A Tool for Designing Re-usable Process in Educational Software Development in Cooperative Environments

Garcia I. and Pacheco C.

Postgraduate Division, Technological University of the Mixtec Region
Huajuapan de León, Oaxaca (Mexico) *www.utm.mx*
ivan@mixteco.utm.mx, leninca@mixteco.utm.mx

Abstract. In the last few years, Educational Software has developed enormously, but a large part of this has been badly organized and poorly documented. Recent advances in the software technology can promote the cooperative learning that is a teaching strategy in which small teams, each composed by students of different levels of ability, use different learning activities to improve their understanding of a subject. How can we design Educational Software if we never learnt how to do it? This paper describes how the Technological University of the Mixtec Region is using a cooperative application to improve the quality of education offered to its students in the Educational Software design.

Keywords. Software reuse, educational software, cooperative learning, process reuse, process notation, process tailoring.

1 Introduction

As education and technology combine, the opportunities for teaching and learning are ever growing. However, the very rapid rate of change in the field of technology poses special problems for academic institutions, specifically for the engineering disciplines.

Nowadays, the software engineering and modern theories of learning converge in the construction of Educational Software (ES) to develop tools that define and implement educational objectives while preserving quality patterns. However, there are still gaps in our ability to assess whether a component or process meets the specified requirements or expectations and needs of students or groups of students. It is wrong to think that ES is less complex than commercial software, which has definitely received more attention from the field of software engineering.

ES covers a range of sub domains, types of systems, requirements and diverse idiosyncrasies which have been covered by the application of principles, specific methods and tools of software engineering in a specialized field. According to Van Schaik, *"clear and unambiguous emphasis on human learning and knowledge acquisition, differ from the educational software with other types of software"* [3]. Thus, ES is evidence of the technological impact on educational processes that has taken

Advances in Computer Science and Engineering.
Research in Computing Science 42, 2009, pp. 53-64

Received 06/03/09
Accepted 27/04/09
Final version 28/04/09

place in recent years, providing a valid alternative to students through an environment of generation (and regeneration) of knowledge.

From our perspective, ES development should incorporate a practical mechanism for designing and establishing effective activities for Instructional Design. The major objective is to facilitate and ensure the accomplishment of educational needs to a target audience, not forgetting to take into account its profile to edit the actual contents. It should also involve users identifying needs and/or specific problems and establishing mechanisms to provide adequate solid educational, communicative and computational principles [4]. However, a basic problem faced by the learning community is to determine how to develop and deliver quality content for learning experiences, while being able to compose, revise and update this content in an efficient way. This brings up the issue of reusability (content developed in one context being transferable to another context).

The profit of a high level of process reusability in software development is a sign of maturity in any discipline of engineering. Software engineering is no exception; in recent years reuse-based paradigms have contributed to the reduction of costs and schedules in the software industry. It is clear that in software engineering the compositional paradigm or component-based paradigm has dominated [20] [11] [17] [7] [10], but in the concrete case of ES, this situation is the opposite; and traditionally, the generative paradigm has been preserved [1] [13]. It is possible that reusability provides tangible benefits, but it is necessary to modify the actual process models in order to incorporate new activities, regarding to the compilation and coherent and centralized information maintenance of a specific domain.

Otherwise, we would stagnate in an ad hoc model where reusability would not generate more than meager benefits. One activity that has significant relevance in a model based on component reusability for software development is analysis and domain modeling. Domain analysis provides, among other products, a definition and a model itself that includes object identification, and common operations and relations among them. But, how to model this domain in an efficient way? The knowledge domain exists independently of the learner, and understanding is coming to know that which already exists. This knowledge can be learned, tested, and applied more or less independently of particular contexts. The use of software technology to support engaged learning goes hand in hand with the constructivism philosophy.

2 Cooperative Learning for Defining Reusable ES Processes

One method cited for accelerating process improvement in ES development is to replicate a standard, reusable process within other projects. However, the creation of a process that is usable on diverse projects is a difficult task. What is needed is an effective method to capture the common and variant elements of project-specific processes and create process definitions that can be applied in a variety of situations, i.e. that are reusable. But what is a reusable process? Feiler and Humphrey defined a process as *"a set of partially ordered steps intended to reach a goal"* [6].

A few years later, Hollenbach and Frakes defined process reuse as *"the usage of one process description in the creation of another process description. It is not multi-*

ple executions of the same process on a given project" [8]. These definitions could be applied to the educational systems context to create a type of graphical repository to produce and refine reusable processes and improve the quality of the final product at the same time.

A literature search shows that groundwork for SE process reuse exists. Research by Sheremetov et. al. [19] proposed the use of agents to expand the simple specification sequence of Instructional Management Systems integrated to SCORM v1.3. The research work of [21] conceptually describes modularity (granularity) of learning sequences, learning activities and actions, reusable learning objects and atoms, and reusable information atoms. Research by [15] relates a special type of labeled material called Intelligent Reusable Learning Components Object Oriented (IRLCOO), producing learning materials with interface and functionality standardized rich in multimedia, interactivity and feedback. In [2], Canales et. al. resumed the development of a Web-based Education System architecture that considers a diversity of requirements and provides the needed functionalities based on creating reusable components for content and evaluation tasks to reduce the complexity, change management, and reuse of learning.

We can observe that the creation of reusable processes is dependent on domain analysis. In spite of existing studies, not many domain analysis methods/tools for creating re-usable ES trough experiences of cooperative learning have been developed [14]. So, this paper presents a systematic and standard method for process reuse in educational systems projects using the "Learning by Doing" as strategy. The purpose of the process definition method is to create reusable ES processes within a repository which can tailor them to specific instructional and technical requirements in a cost effective manner. Reusable processes aid in transferring process knowledge between projects, reducing instructional costs, reducing effort, planning common educational projects and activities based on process data, and increasing quality in a continuously improving process oriented environment. The ES process life cycle contains the following steps:

- *Define reusable ES process from repository.* Even if a previous ES process does not exist, the student must start from here. The output is of one or more process descriptions, along with tailoring guidance and output products (see Process Manager in Section 3.1.2). The process descriptions are also integrated into the repository if they have been tested to ensure they are fit to (re)use.
- *Select process to adopt in new project.* Once the re-usable ES process is complete, the Process Manager is ready to deploy it in a real environment.
- *Tailor the ES reusable process on a project.* The re-usable ES process is tailored to meet the specific technical and instructional requirements and environment of an educational project.
- *Enact the process on the project.* The process is put into practice on the educational project. An assessment function evaluates the educational project to ensure that the new process is faithfully enacted.
- *Refine the process.* Based on the previous evaluation the process definition is then refined and inserted in the repository as a "good" process.

3 Building an Alternative Repository of ES Processes

A learning approach to ES development captures project-related information during the creation of individual ES artifacts which is then disseminated to subsequent educational projects to provide experience-based knowledge of development issues encountered at a repository of effective processes. These principles have been successfully applied in software engineering in [8] and [9], and we are trying to apply them in the ES context through an exploratory prototype, named ESPLib (Educational Software Process Library).

The initial ESPLib research focused on developing a tool to support the creation of domain repositories. In particular, we focused on defining precise processes to accomplish the instructional design. We assume the hypothesis that the most difficult aspect to students designing ES is the identification of the correct learning domain. They could be great programmers, capable and efficient, but they may not know how to represent in clear terms how to design the learning environment. ESPLib is composed of domains which are independent knowledge realms consisting of a set of activities and rules (reusable process) that define the context in which an ES process is applicable to a learning objective.

Our approach, illustrated in Figure 1, combines a rule-based system to match project characteristics to ESPLib processes and a deviation process to continuously update and improve new practices. At key points in a development procedure, students are taken through a set of questions designed to elicit educational project characteristics and match them to specific process elements in our schemas (or lifecycle models). This creates a customized process that goes through a review procedure validating the path taken through the rule-based system and assessing whether the processes assigned to the ES project are necessary and consistent, or are in need of further refinement or correction. The result is a modified process which should be followed.

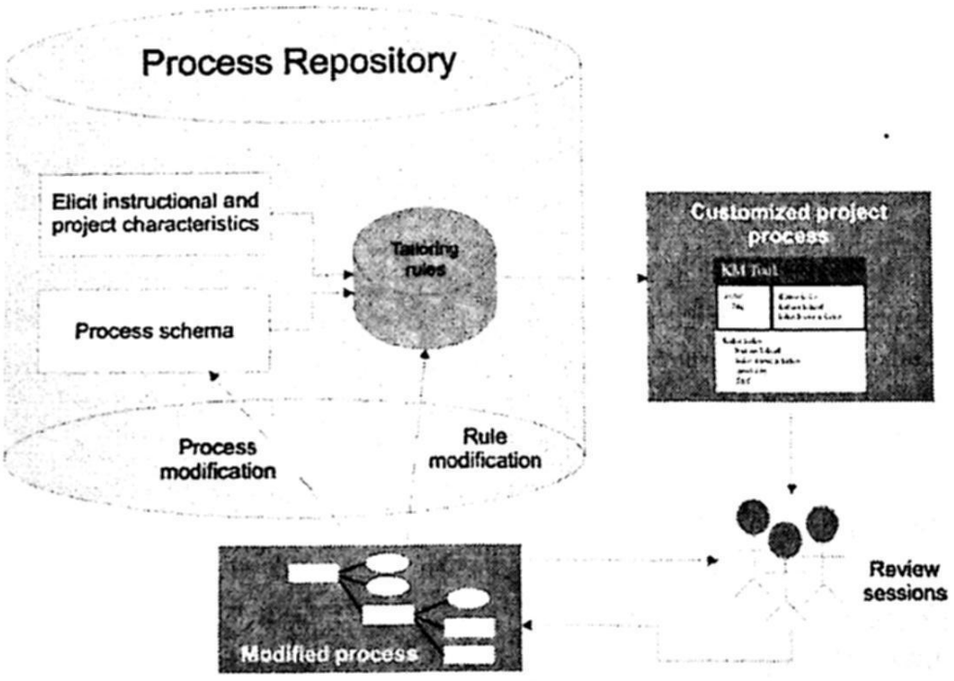

Figure 1. Tailoring and modifying ES process.

Cases are created to track conformance and document the development process for that ES project. In instances where deviations from the assigned process are requested, students must decide whether changes to rules and/or process elements are needed. In other words, a deviation from the current standard (defined by the rules and processes) sets a precedent that defines future actions under those circumstances. Thus it is clear that ESPLib implements the constructive theory of Jean Piaget: *you can learn if you can do it* [16].

The use of software technology can help the constructivist theory because it is possible to build a "Learning by Doing" environment that also combines the constructivist approach and the cooperative learning within a process and practice repository. The five elements in [5] define the cooperative learning: positive interdependence, individual accountability, face-to-face primitive interaction, appropriate use of collaborative skills, and group processing.

3.1 The ESPLib Tool

The main interfaces for ESPLib are shown in Figure 2. The Project Manager, shown to the left in Figure 2, displays a hierarchical arrangement of educational project activities.

Figure 2. ESPLib Project Manager.

In the figure, a project named "English Lab System" has been chosen from the list of current projects. Each activity contains project-specific information, as shown in the window on the right of Figure 2, which was obtained by double-clicking on the activity named "Prepare for Educational Design" in the project hierarchy.

Processes are used in a rules-based decision support approach, and describe specific guides to establish development activities in ES projects. Guides consist of a de-

scription of a specific activity, the related activities, a description of input and output, and specific information about process notation. For example, in Figure 2, the process describes the specific activities of preparing educational design. This description was updated when the process was added to the repository of ESPLib

3.1.1 Creating Domains

As we said previously, ESPLib is composed of domains. Domains are independent knowledge realms that consist of a set of domain diagrams defining activities and domain conditions that define the context in which a process is applicable to a specific learning objective. Currently, educational projects belong to a single domain, which is chosen when a project is created in ESPLib. All subsequent project activities will use the activities and notations defined for that domain.

A domain defines the area of possible activities for educational projects within a given field (e.g. computer science, mathematics courses, electronic practices, learning a foreign language, etc.), structured in a work breakdown plan. They define standard activities that have proved useful for situations encountered by educational projects within that specific domain. The diagrams describe some of the problems or necessary steps that must be taken to ensure that the procedure is correctly followed, and can be updated by a project to reflect specific activities that this project follows.

The obtained descriptions are used to tailor the ES development process to the specific needs of different projects. Internally, ESPLib uses a simple forward chaining production mechanism implemented using an SQL database to represent alternative processes. The selection of a domain depends on a set of preconditions and events. Preconditions are represented by question/answer pairs. Questions are associated with the Questions tab of the Open or Add process Options. Questions are chosen by students to tailor the ESPLib's standard process to individual educational needs. These options address high-level project requirements which may influence which educational activities are chosen for the project.

When all preconditions of a domain evaluate to true, the Domain Manager "fires", causing a set of defined actions to be executed. The core actions in ESPLib can remove questions from the question stack, add a question to the question stack, or add a domain to a project. In addition, new diagram types can be created by using the Tool Option. Once the domain is identified, the student must describe and define the development process in formal terms.

With this multiple-choice questions ESPLib control the student's knowledge, on the processes repository, on the data database, and on the theory of software engineering.

3.1.2 Defining Notation

Process definitions are specified using the process definition component of the Process Manager and are based on a defined framework using notation.

Nowadays, in the context of ES, the research lines are focused on working with the specific components of systems, like content. There is an initiative that inculcates globalization of materials for its use in different learning programs and sessions through the use of "metadata". In this category, the proposal that most stands out is the Learning-Object Model (LOM) developed by IEEE-LTSC.

This model is composed of nine different records that identify general characteristics, lifecycles, metadata, technical and educational issues, relations with objects, and classification of resources [12]. ESPLib combines the LOM approach with the notation from Object Modeling Technique (OMT). OMT is one of the methodologies oriented to object analysis and design which is more mature and efficient than currently exists [18].

The great advantage of this methodology is its open nature (non-proprietary), which allows it to be in the public domain. This characteristic facilitates its evolution to match all current and future educational software needs. Process definitions are extended to include educational project information using the Process Manager component. Interactions between students and ESPLib occur through a graphical user interface (GUI). ESPLib's GUI consists of a main window containing a process display area, various pop-up dialogue windows which can be dismissed when no longer required, and a toolbox that can be displayed or hidden as required by selecting the Tool option. The GUI is illustrated in Figure 3 and is an example of an ESPLib's loaded process.

A process model must be defined using ESPLib before it can be saved. Process models are defined using the graphical notation and are constructed from a set of basic process elements which are displayed in the toolbox, shown to the right in Figure 3. ESPLib provides a notation that groups "process elements" in four types: phases, activities, transitions and documents.

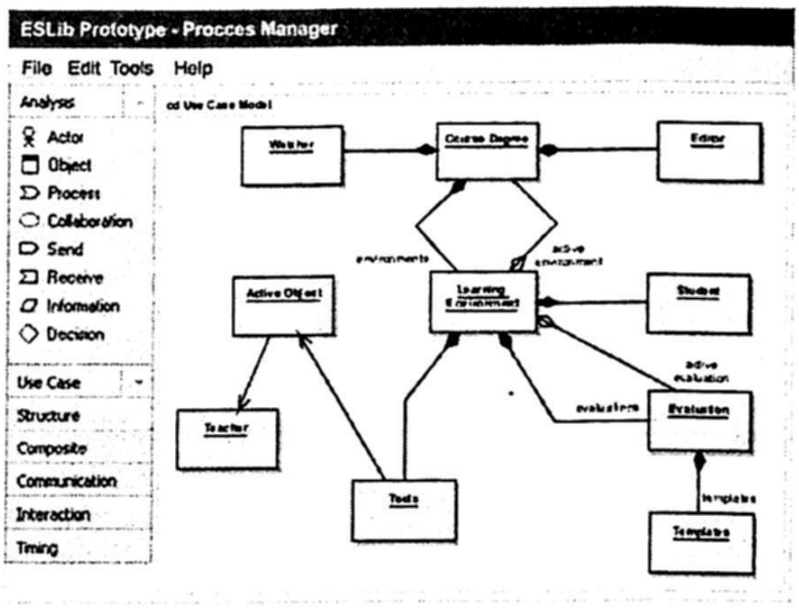

Figure 3. An Example of ES process in ESPLib.

A phase is a process element that is used to group other process elements, and can be used to build a hierarchical educational process definition by encapsulating other phases. It groups related activities together into a single unit and is often used as a synchronization mechanism. ESPLib offers two types of phases: a normal phase and a decision phase, called Phase and Decision respectively, in Figure 4.

A decision phase differs from a normal phase in that a Yes/No decision must be made as a result of undertaking this phase. A normal phase has no such requirement. An activity is a low level element which cannot be broken down further.

Figure 4. ESPLib notation.

A document process element is used to represent any artifact which is produced, including a source code and class documents, during the learning process. The final type of process element is a transition of which two types exist –a single directional arrow, called Movement, and a bi-directional arrow, called a Double Movement. Both types of transitions act as a connector between the elements of the process, with the transition from one process element to the next caused when the first is completed.

The single directional arrow is used to represent flow in one direction only towards the arrow head. The bidirectional arrow behaves the same as a single directional arrow in that information flows to the bold arrow head.

The difference between these two process elements is that a bidirectional arrow permits backtracking through a process if necessary. This could be triggered by omitting a learning objective or requirement in the previous analysis. A bidirectional arrow will allow an earlier phase to be re-executed to fix a problem. This arrow allows process definitions to be less cluttered than if explicit decision phases were required after every normal phase to determine if the previous phases were implemented correctly or if problems were identified. Each element has an expected duration and an indication of who is responsible for the execution of the process element. In addition to this, a document process element also contains the name of the document and where it is located in the learning process.

3.2 Creating and Modifying ESPLib's Processes

To define an ES process for use on a project, ESPLib allows an existing process model to be modified. Alternatively, the process can be defined starting with an empty process. Only one student of ESPLib is permitted to modify a software process at any one time, thus removing the possibility of multiple concurrent updates to the ES process definitions.

Process elements can be added to a process by selecting the required component in the toolbox and then drawing it into the process display using the mouse. Elements can be removed from a process by selecting the element and then choosing "cut" from the edit menu. When any modifications are made to a process the student is required to justify why the change was made. This is designed to provide some reasoning for the ES process that is to be followed for a particular project. The justification document for the process contains all the modifications, who made them, when they were made and the reason for the change. This file can be viewed by any student at any time.

Process models in ESPLib are defined hierarchically through the use of phases, reflecting the fact that a process consists of a series of levels. In ESPLib, it is possible to look inside a phase to see the elements that it contains, thus allowing activities in lower levels of the process hierarchy to be defined. Processes that do not conform to the defined process framework, which sets out rules which each process must abide by, are flagged as incomplete and may not be followed until they conform to the framework. The process framework is designed to assist in the description and modeling of educational software and is defined as part of ESPLib. This allows a process to be developed over a period of time. Storing process definitions allows previous models to be used as the basis for the construction of new process models.

ESPLib supports the dynamic modification of process definitions. This allows process instance to evolve throughout the lifetime of the educational project. This is especially important for projects of long duration. When a process definition is modified whilst it is being executed, the changes made to the process must still conform to the defined process framework. An example of a new process created in ESPLib is shown in Figure 5. Our students created a process to develop ES and help with foreign language classes in the English Lab. They modified an existing process (DIS12) and created an alternative process, the DIS13.

The Learning Environment phase was added because teachers considered that students might use computers, tapes, movies and books. The ESPLib's repository includes all the assets (templates and documents) collected to analyze a complete ES.

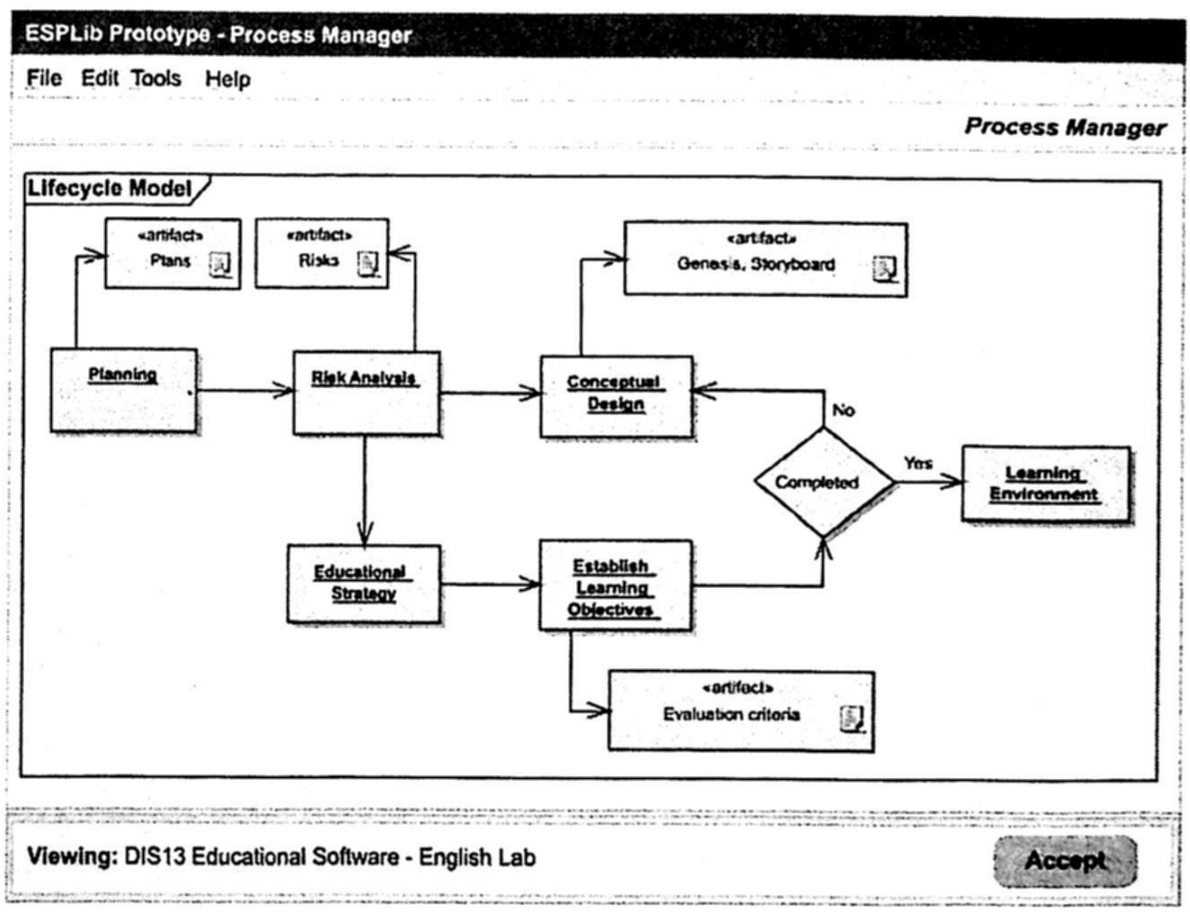

Figure 5. An ESPLib's process.

4 Evaluation and Future Work

As mentioned before, a tool for designing ES does not exist. There is a lot of information on how to improve the quality and performance of ES, but ESPLib attempts to "implement" that knowledge, in a real tool, to avoid the repetition of tasks already developed and typical errors. Traditional education is based upon a paradigm normally called the "knowledge reproduction model". This model is based on verbal lecture, drill and practice sessions, printed handouts, structured classroom activities, and office hours. In its pure form this model is grounded in the belief that knowledge is objective and the purpose of the teaching process is to transfer this static body of knowledge from its source to the student. The student, using this point of view, is seen as a passive learner "waiting to be filled" with knowledge, but the knowledge is not static. Learners have to actively interact with the learning environment and contents by browsing, searching, selecting, scanning and so on.

The general goal of ESPLib is to create educational knowledge management tools that are more proactive in delivering information to students than typical repositories. One way this can be accomplished is to create a tailorable process that provides domain-sensitive information to ES development efforts. The ESPLib tool and framework is flexible enough to bridge the gap between overly-restrictive ES development methodologies and ad-hoc practices to fit the needs of ES as they evolve.

The small number of process elements available for use in a process definition assists students understanding. Although only a small number of process elements exist, it is still possible to construct a process definition, such as the English Lab process, without the definition becoming overly cluttered.

It is possible in ESPLib to add additional attributes to a process element or to customize process elements. This is a valuable feature in a process modeling tool, allowing process definitions to be customized to the style used by any student. ESPLib allows dynamic modification of a process definition. This is a very valuable feature as process definitions are dynamic in nature and may change throughout the lifetime of a project. Another positive feature of ESPLib's process definitions is that they are easy to understand by all students due to the diagram notations. The process framework is not currently customizable making it impossible for any student to specialize the process rules to incorporate their specific policies. Storing the process definitions allows past process models to be examined and evaluated. This can be valuable when a process model is being chosen for use on an educational project because a successful process can be chosen.

It was clear from our brief study that the domain interface of ESPLib (Domain Manager) needs some improvements. In particular, there needs to be a method of finding answers to some behaviors (such as a case being added to a project) or other attributes.

Currently, students need to page through the questions one-by-one to find an answer with the desired precondition or action. Another improvement is related to the addition of a "content editor" module, the main idea being to relate each phase to a topic from the normal course and include (or modify) it according to the students needs.

5 Conclusions

Knowledge management for educational software development is more than just repositories and models. It also actively requires work between pedagogue and software specialists delivering information during the development process and ongoing process of capturing project experiences.

Using the knowledge management techniques can prevent the duplication of efforts, avoid repeating common mistakes, and help streamline the development process. The reviewed approaches have shown that the reuse process provides relevant useful and up-to-date information to improve the quality of ES. This mandates a strong tie between technologies and pedagogy in which using the technology must become part of routine work activities.

The contribution of this research pretends to define and implement a single tool to improve the analysis and design of ES using the cooperative learning approach, with the intention of successfully merging pedagogical and technical aspects equitably. As the repository of ESPLib grows through the principled evolution of the knowledge domain, it also becomes able to handle a wider range of domains, while evolving towards answers to problems that fit the student's technical and pedagogical context. The real question is not whether the repository of ESPLib is "correct" in some objective sense, but rather whether less mistakes are repeated and better ES solutions adopted when using the repository. Students who will work with the ESPLib' cooperative learning environment and who will have opportunities to work cooperatively with students who have different ability, ethnicity, gender, and so forth will be better able to build positively interdependent relationships than students who will have only an individualistic and a competitive learning.

References

1. Biggerstaff, T. "A perspective of Generative Reuse" Annals of Software Engineering. 5:169-226. 1998.
2. Canales, A., Peña, A., Peredo, R., Sosa, H., & Gutierrez, A. "Adaptive and intelligent web based education system: Towards an integral architecture and framework" Expert Systems with Applications 33: 1076–1089. 2007.
3. De Diana, I y Van Schaik, P. "Courseware Engineering Outlined: An overview of some research segues". ETTI 30(3): 191-211. 1993.
4. Díaz, M., Pérez, M., Grimmán, A. & Mendoza, Luis. "Proposal for Development of a Methodology for Educational Software under a systemic quality approach". Universidad Simón Bolivar. Venezuela. 2005. (in Spanish).
5. Felder, R. M. & Brent, R. "Cooperative Learning in Technical Courses: Procedures, Pitfalls, and Payoffs" ERIC Document Reproduction Service, ED 377038, 1994.
6. Feiler, P. & Humphrey, W. "Software process development and enactment: Concepts and Definitions" Software Engineering Institute CMU/SEI-92-TR-04. Pittsburgh, PA. September 1992.
7. Fiorini, S., Leite, J., & Lucena, C. "Process Reuse Architecture" Proceedings of the 13th International Conference CAISE 2001. Advanced Information Systems Engineering, Lecture Notes in Computer Science 2068:284-298. 2001.

8. Henninger, S., Lappala, K. & Raghavendran, A. "An Organizational Learning Approach to Domain Analysis", Proceeding of the 17th International Conference on Software Engineering, 95-104. 1995.
9. Henninger, S. "Case-Based Knowledge Management Tools for Software Development", Journal of Automated Software Engineering, 4: 319-340. 1997.
10. Henninger, S. "Tool Support for Experience-Based Methodologies" Proceedings of the 4th International Workshop on Learning Software Organizations, Lecture Notes in Computer Science 2640. 2003.
11. Hutchens, K., Oudshoorn, M. & Maciunas, K. "Web-Based Software Engineering Process Management" Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences (HICSS), 1: 676. 1997.
12. IEEE/LTSC. Institute of Electrical and Electronic Engineers, Inc./Learning Object Model. URL: http://www.ieee.org/lom
13. Okamoto, T. "The Model of Collaborative Learning and Technological Environment for Evoking Interactivity-Building of Knowledge". Proceedings of the IASTED International Conference, Computers and Advanced Technology in Education; Rhodes, Greece, 2003.
14. Peña, A., Sossa, J. "Web-based Education: A state of the art" UPIICSA and Computer Research Center IPN. 2005. (in Spanish).
15. Peredo, R., Ocaña, L. & Sheremetov, L. "Development of intelligent reusable learning objects for web-based education systems" Expert Systems with Applications. 28: 273-283. 2005.
16. Piaget, J. The Psychology of Intelligence. London Routledge. 2001.
17. Reis, R., Lima Reis, C. & Nunes, D. "APSEE-Reuse: A Case-Based Reasoning Model for Reuse and Classification of Software Process Assets" Proceedings of 7th International Workshop on Groupware (CRIWG'01). 2001.
18. Rumbaugh, J., Blaha, M., Premerlani, W. & Eddy, F. Modeling & Design Object Oriented. Prentice Hall. 1997
19. Sheremetov, L. & Peredo, R. "Development of Reusable Learning Materials for WBE using Intelligent Components & Agents", Technical Report, Instituto Mexicano del Petróleo and Laboratorio de Agentes del Centro de Investigación en Computación, México. 2002. (in Spanish).
20. Succi, G., Benedicenti, L., Predonzazi, P. & Vernazza, T. "Standardizing the Reuse of Software Processes" StandardView, 5(2): 74-83. June 1997.
21. Vladimir, U. & Maria, U. "Reusable learning objects approach to web-based education" International Journal of Computer and Applications, 25(3). 2003.