Interdisciplinary Journal of E-Learning and Learning Objects **Volume 4, 2008**

Formerly the Interdisciplinary Journal of Knowledge and Learning Objects

A Systems Engineering Analysis Method for the Development of Reusable Computer-Supported Learning Systems

David Díez, Camino Fernández, and Juan Manuel Dodero
DEI Laboratory – Computer Science Department
Universidad Carlos III de Madrid, Spain

david.diez@uc3m.es
camino.fernandez@uc3m.es juanmanuel.dodero@uc3m.es

Abstract

The development of computer-supported learning systems is a complex task that must take into account diverse issues and perspectives. Given the difficulties attached to this process, different authors have proposed the use of software engineering as a reference to optimize the courseware development process. Following this trend, and given the efficiency of the outcomes, it seems convenient to adapt existing product line development principles from software engineering to the development of learning resources. The purpose of this paper is to postulate a specific analysis method for computer-supported learning systems that guarantees the quality of the development. This specific analysis method can facilitate the learning object reusability and minimize the relevance of the expertise in the development of learning resources.

Keywords: Computer-supported learning systems, reusability, analysis method, learning objects.

Introduction

Reusability is considered as an essential and undoubtedly the most important quality of learning objects (Sicilia & Garcia, 2003). A *learning object* is defined as follows (McGreal, 2004):

> *"A learning object is any reusable digital resource that is encapsulated in a lesson or assemblage of lessons grouped in units, modules, courses, and even programs. A lesson can be defined as a piece of instruction, normally including a learning purpose."*

So far, there have been many attempts to support learning objects reuse, but most of these efforts have focused on defining reusable patterns (Baggetun, Rusman, & Poggi, 2004; Jones & Boyle, 2007) and on designing artifacts for evaluating and recovering materials (Cuadrado & Sicilia, 2005; Padrón, Díaz, & Aedo, 2007). Therefore, in keeping with the experience in software development, effort should be put to define methods that permit the discovery of object commonalities. In turn, this information would allow for reuse.

Since reusability refers to prospective and future usage scenarios, it is difficult to manage. The concept of reusability encompasses aspects related to format, interpretation, and pedagogical suitability (Sicilia, 2004). Thus, in order to achieve effective reusability it

Editor: Janice Whatley

is necessary to tackle the problem from the point of view of:

- The learning objects. The correct definition and formalization of learning objects to allow for determining their potential reuse.
- The learning contexts. The adequate specification of the learning context permits determining the appropriate instruction method and, thereby, identifying the more reusable tasks or activities.

The standards and tools previously available have focused on the first of the views presented above by dealing with aspects regarding format (but which overlooks the analysis of the specific context). In order to overcome this matter, the study of reusability in the development process of computer-supported learning systems has been proposed. A computer-supported learning system can be considered as a structured set of learning components, whose qualities and characteristics rely on the learning components that compose it. Thus, according to this notion, reusability refers to the ability to use services or learning objects for multiple courses or lessons (Hartshorne, 2003). In this context, the computer-supported learning systems reusability must take into account the features of the learning system to identify, recover, and use the learning components that comprise the system. Based on the experience of software development, reusability should be considered at the early phases of the development process (Kang, 1998). For this reason, and with the purpose of facilitating the identification of commonalities and further reuse, the definition of systematic analysis methods is needed.

The rest of the paper is organized as follows. The second section describes the motivations for a specific analysis approach in the development of computer-supported learning systems. The following sections define our analysis method and include an example on how the approach has been applied. Finally, a set of conclusions and recommendations for future work are presented.

The Motivation for a Specific Analysis Method

Courseware is defined as any instructional system delivering content via computers that supports learners as well as teachers in their educational efforts, in a technical and instructional way (Grützner, Ruhe & Pfahl, 2002). The life-cycle of the earlier courseware development methods resembles the phases of the traditional software development process: analysis, design, development, and evaluation. As Goodyear points out, the relation between software engineering and learning resources development is particularly strong in analysis-related tasks:

> *"The main areas of overlap between software engineering and courseware engineering are probably to be found in those areas concerned with requirements analysis and design."* (Goodyear, 1995)

The analysis phase is essential in courseware development processes (Wiegers, 2003), as well as being one of the determinants of product success (Gagne & Medsker, 1995; Hadjerrouit, 2007). These qualities are themselves very meaningful; nevertheless, we can identify two additional factors that make analysis one of the most relevant activities in the development of computer-supported learning resources: (1) Effective reuse must be planned and considered early in the development life cycle. (2) Reusability of computer-supported learning systems depends on using and reusing correct learning components to facilitate a concrete learning context.

Taking these judgments as a main starting point, we focus on the revision of the analysis stage. The aim of this section is to provide a better understanding of the motivations that lead to the definition of a specific analysis method.

The Utility of Systems Engineering Methods

Instructional software development models were conceived as patterns that adapt to the automation process of the instruction and guide its development. The result is a set of models that focusing on the instruction process and aim at improving the quality of the products. However, despite the efforts made and the existence of models widely used, there are no absolutely satisfactory approximations (Grützner et al., 2002; Hadjerrouit, 2007). The review of the different software development methods for instruction has shown that there are two main causes for the reduced usefulness of such methods, namely:

- The almost direct relation between software development methods and instructional design models. These are models that have incorporated ideas from other disciplines in order to ease the automation, but they are based on common instructional design processes. This strand of work complicates the development of computer-based systems.
- Its final use: a process model is not necessarily useful for any situation or context. The search for general methods reduces the precision, completeness, and efficiency of such models. However, all the development models heretofore reviewed are conceived as 'general' models.

In an attempt to overcome these barriers, it is crucial to apply specific methods to the development of computer-supported learning systems. By adapting themselves to the characteristics of the context, these specific methods must be able to integrate the principles of instructional design with those of software engineering.

The Function of the Analysis Stage

The analysis stage is the front-end phase of the development process of computer-supported learning systems. This phase constitutes an essential step of the development process and one of the critical issues that determines the quality of the final product:

> *"The analysis phase sets the stage for the whole project. The necessary groundwork for understanding what the project is all about is completed in this phase. We take the strong position that the more effort you put into planning, the smoother the rest of the project will go and the better the quality will be of your final product."* (Alessi & Trollip, 2001)

In spite of its relevance, there is not a homogeneous view on the usefulness or functionality of the analysis stage in the development process:

- *Sequential models*. The models define a sequence of stages, in which the results of each stage are the input for the subsequent one. The sequential models are derived from instructional design models. The MISA method (Paquette, 2004), the 'Alessi & Trollip' method (Alessi & Trollip, 1991) and the 'Dick & Cary' (Dick & Cary, 1990) are included within this category. The sequential models conceive the analysis as a phase used for the definition of the educational scenario: objectives of the instruction, profile of the student, learning environment, etc. The goal of the analysis is to define the educational context in which such a system is meant to be deployed.
- *Iterative models*. The development of the system is done by progressively refining the characteristics of the system. The iterative models are derived from software engineering principles and artifacts. Amongst the iterative models, rapid prototyping is becoming a dominant means of design and development (Goodyear, 1997). The knowledge iterative models propose an analysis phase focused on contents and the presentation style.

With the purpose of overcoming these barriers and setting the ground for an analysis method, it is essential to establish the principles that the analysis stage must satisfy for this type of system:

- The instructional domain of an educational problem must be represented and understood. The scope of the instructional domain must be studied, defined and represented.
- The instructional system should be specified. The behavior of the system, its needs and conditions, must be identified and represented.
- The domain and system information must be arranged, documented and represented in order to avoid unnecessary and worthless details.

If borne in mind at the analysis stage, these principles will help to have a better appreciation of both the instructional and computational aspects of the system.

An Approach to Instructional Engineering Analysis

Having reviewed the problem, we concentrate on the establishment of mechanisms that might be useful to solve it. This section introduces our proposed solution: a specific method for analyzing computer-supported instructional systems. First of all, the method requirements will be enumerated; secondly, taking into account the review of the context, we explain the principles which our method is based on. Finally, we present an overview of our method.

Method Features

Analysis is the stage by which the needs and conditions of the problem are determined in order to specify the characteristics of the system under development. In order to accomplish this objective, the method here proposed fulfills the following features:

- Complete. The method must deal with the specification of the problem in all its dimensions. It must avoid any possible omissions that can lead to misunderstanding of the problem. As a result, the method must include the study of instructional attributes and the computational aspects of the system.
- Prescriptive. A method establishes the most suitable set of activities that should be carried out so as to achieve a purpose. The method must ensure the provision of an analysis method that guides the specification of the system.
- Consistent. The method must avoid the contradictions that might arise between the different activities. The result of the specification of the system must be consistent with the context of the application and the requirements of users.
- Systematic. The method must provide similar results in similar contexts. The definition of the method artifacts must be carried out concisely and rigorously so as to guarantee its efficacy, independently of the participants.

An appropriate analysis of the system depends on the mechanisms employed for the representation of knowledge. Such mechanisms are significant to avoid confusion, doubts, or misinterpretations and, therefore, they determine the success of the system. In this regard the analysis method proposed must provide artifacts that, from a certain degree of formalization, allow reducing ambiguity and vagueness. Finally, in order to set the basis for future applications, the method must set guidelines that facilitate the automation of the analysis process.

Engineering Principles

A specific method for analyzing computer-supported learning systems should take into account informational technologies and, accurately, software engineering principles. However, not all software engineering paradigms can be useful in the field of instructional design and, further, translating partial ideas from one discipline to another is not recommended. According to this

thought, the proposed method is based on Product Line Engineering (Clements, Northrop, & Northrop, 2003) and, specifically, on Domain Engineering.

Domain Engineering addresses the creation of domain models. A domain model represents the set of requirements that are common to systems within a product line or context (Prieto-Diaz, 1990). Domain Engineering proposes a development model divided into three phases. The first of such phases, the domain analysis, is the object of interest of this work. Domain analysis has been selected as the development paradigm because:

- Courseware development needs a structured evolutionary process model in order to deal with change and evolution (Hadjerrouit, 2007). Domain analysis proposes a continuous evolution of the domain model, and development of a particular system that exploits previously accumulated domain knowledge can be the source for new insights about the domain that adds to or refines codified domain knowledge.
- Domain analysis helps to build reusable components (Prieto-Diaz, 1990). Reuse is an essential characteristic of designing learning objects and, consequently, of developing computer-supported learning systems. To ensure the reusability of the domain models produced, domain analysts use diverse sources of domain knowledge. These sources provide information on the range of potential components and services in the domain.

Domain analysis presents a use-related restriction, as it is often applied on stable and structured domains (Arango, 1994; Czarnecki, 1998), with rigorous information sources that allow defining the knowledge of the context in a domain model. Thus, and prior to describing our analysis method, two issues should be explained: the concept of domain and the information sources that can be established in the educational context.

The domain definition

The first aspect to be defined concerns the domain and its scope. A learning domain is defined as:

> *"The set of educational scenarios that share contents, an area of work, as well as pedagogical objectives and a specific instructional form. The magnitude of the educational contents considered must make sense by themselves".*

The definition here proposed follows the typical view of the domain analysis methodology: the concept of domain is related to the type of problem to be solved; that is to say the domain is determined by the set of systems that satisfy a common functionality (Ferré & Vegas, 1999). In the case of instructional systems, the functionality to be solved is understood as the systematization of the learning process (Tennyson, 1995). Given that the learning process depends on the instructional form (which is in turn conditioned by the contents, the context, and the pedagogical objectives), these factors have been taken as essential for the demarcation of a domain. Regarding scope, the domain considered must go beyond the domain of a certain course or subject and it must be more reduced than the instructional model or the group of subjects taught within a particular field of knowledge.

The sources of domain information

Instructional design emerged last century in order to improve the teaching and the learning processes. Instructional design is a knowledge area oriented to having a better understanding of, as well as explaining, the instructional process. With this purpose, instructional design determines the means by which a specific educational objective can be achieved. The application of instructional design aims to suggest methods and models that better describe the instructional task (Dijkstra & Merriënboer, 1997). Instructional design provides formalized knowledge, which can be used as a source of information for the analysis of educational domains. In addition, the cata-

logues and models of good practices, presented in the IMS-Learning Design (IMS-LD) specification (Koper, 2005), can be taken into account, as well as the expertise of the domain and existing development models and instructional resources already developed. In conclusion, we can identify sufficient and valid information sources so as to show the existence of a stable domain.

The Learning Analysis Method

Learning Analysis is conceived as an evolutionary instructional engineering analysis method that enables improving the reusability of learning components. The method is based on a set of specific artifacts and an analysis process that follows the domain analysis ideas.

Engineering artifacts

An essential requisite of an analysis method, since it establishes the conditions and needs of the system under development, is to avoid ambiguity and vagueness. To this end, it is necessary to make use of suitable artifacts that allow the specification of the system. The artifacts deployed must help to represent the domain model and the system model. In relation to the domain model, several software engineering approaches propose a careful analysis of the domain; however, feature modeling is one of the most effective techniques to represent the domain knowledge (Kang, 1998). The underlying motivations for using feature modeling are:

- Users, instructional designers and application engineers usually communicate in terms of application features. These terms in a domain constitute domain terminology and they are used to characterize specific applications (Lee, Kang, Chae, & Choi., 2000).
- Reusable software contains inherently more variability than specific applications, and feature modeling is the key technique for identifying and capturing commonality and variability (Czarnecki, 1998).

Feature modeling captures the capabilities of an application in a domain in terms of *features*. A feature model represents the common and the variable features of the concepts involved in the problem and the dependencies between them. In our approach, a feature is defined as:

> *"A distinguishable characteristic of a system concept that is relevant to the instructional process in a specific learning domain."*

A feature model comprises a feature diagram and some additional information, such as a short semantic description of each feature. A feature diagram consists of a set of nodes, corresponding to a domain feature and the commonality level of such feature, such as high or medium, a set of directed edges, and a set of edge decorations. The nodes and the edges form a feature tree. The edge decorations define the relation among the sub-nodes of a particular node. As shown in Figure 1, the edge decorations are drawn as arcs connecting subsets or all of edges originating from the same node. Different types of features concern different types of interests in systems development, so a system cannot be built unless these different groups of features are decided upon by each respective group of experts (Kang, 1998). Especially, we have classified the instructional domain features into four different categories:

- Capability. Features that are associated with the services or functions provided by the system and which constitute the main concern of users: teachers and students. It corresponds to contents, learning services and the attributes of potential students.
- Domain technology. The category represents instructional design details. In an educational context, domain technology features compile the instructional theories and the most suitable methods to the domain. It is the concern of instructional designers.

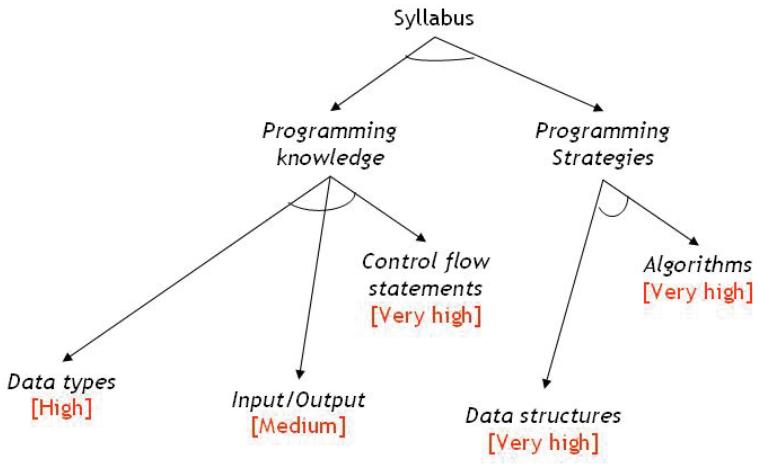

Figure 1: Feature diagram with optional features.

- Operation environment. An operation environment feature represents attributes of the environment in which a system is used. It corresponds to the instructional standards and learning specifications that are the concern of system designers.
- Implementation technique. The category represents implementation details used to develop and deploy a computer-supported learning system. It comprises platforms, tools, learning activities and supported services used in the domain. Implementation technique features are the concern of end developers.

Engineering process

In addition to engineering artifacts, Learning Analysis provides a well-defined engineering process. An engineering process is understood as a networked sequence of activities, tasks and events that embody strategies for accomplishing a product development. In our approach, the analysis process is broken down into two stages:

- Domain analysis. The domain analysis stage consists of activities for studying the domain knowledge and creating a model. The stage aims at analyzing the sources of domain information in order to define a set of commonalities for the systems in such a domain.
- System analysis. Activities for analyzing a specific learning system by using the domain model created in the domain analysis stage. The system analysis aims at specifying the constraints and functional requirements of a specific computer-based learning system.

The use of an analysis method divided into two independent but interrelated processes is one of the reasons for the success of reuse. Domain analysis focuses on supporting systematic and large-scale reuse by capturing both the commonalities and the variability of systems within a domain (Czarnecki, 1998). The results of the domain analysis are used by the system analysis to identify potential elements for reuse in a specific application. The following paragraphs focus on a more detailed description of the activities that make up each of these stages.

Domain analysis. The purpose of 'domain analysis' is to select and define the domain of focus, collect relevant domain information, and integrate such information into a domain model. The final goal is to specify domain learning commonalities in order to identify potential learning components. The domain analysis stage consists of three phases:

> Stage 1: Domain scoping. This phase defines the scope of the domain. Domains are abstraction that group particular sets of systems or areas of learning. Moreover, a domain

can overlap and even enclose other domains. Scoping a domain is therefore not like choosing an item out of a catalog. Keeping these factors in mind, the activities for domain scoping are identified as follows:

- Domain selection is the first activity of the domain scoping phase. Domain engineers perform a scan for possible learning domains, working with the stakeholders to refine their selection and choose the right domain. The choice of domain should be driven by the contents and the pedagogical objectives.
- Having identified the target domain and the domain boundary, the information on the domain needs to be compiled. As explained previously, the information can be provided by experts or collected from available references (instructional design theories and models, best practices and best practice patterns), domain experts and knowledge learning systems.
- Document the context. The specific terms used in the domain are defined in the domain terminology dictionary. The terminology dictionary serves as a communication medium between stakeholders and designers.

Stage 2: Domain featuring. Once the domain has been scoped, the domain featuring phase provides the steps to identify the features of the domain. Domain featuring defines a set of reusable and configurable requirements for specifying the learning system in an instructional domain. Additionally, the name and the commonality level of the identified features must be established.

Feature identification is guided by 'feature starter sets' (Czarnecki, 1998). Feature starter sets define the collection of features suitable for the domain. In our method, such sets of features correspond to the categories enumerated previously: capability, domain technology, operation environment, and implementation technique. For each category, compiled information is reviewed, differences and similarities between domain entities are recorded, and specific features are identified.

Stage 3: Domain modeling. Once the features of the domain have been identified, named, and classified, a hierarchical model should be created by classifying and structuring features. Hierarchical diagrams allow domain engineers to classify features as mandatory, alternative, and optional.

In addition to the feature model, a set of business restrictions rules between features must be defined. Such rules are a type of constraints on the use of a feature. Restriction rules have four outlines: (1) One feature requires the existence of another feature; (2) Two features are mutually exclusive; (3) One feature should be used with another; (4) One feature should be preceded by a set of features.

Whether the domain model correctly represents the features of the domain and the restrictions between them should be validated by domain experts.

The domain model represents the knowledge information of the domain. This is a dynamic model that can be refined continuously over its life-cycle by adding new knowledge. As more knowledge is added to the model, it becomes more precise and useful.

System analysis. 'System analysis' is the process of building learning systems based on the result of the domain analysis. In the same way as domain analysis, this stage consists of three phases:

Stage 1: System scoping. The first activity of the system analysis stage is the definition of the system scope. The scope of the system is laid down by its functionality; that is, by a set of business needs and conditions. In a computer-supported learning system, the scope is determined by two kinds of needs: (1) Instructional need - a collection of learning

goals that will be carried by the system; (2) Computational needs - requirements related with the system behavior. In addition, administrative and organizational conditions should be considered.

Stage 2: System featuring. Once the scope of the system is defined, and taking as a referent the domain model, a selection of the features that match customers' needs is made. The selection of features is achieved by following guidelines:

1. First of all, an effective method for identifying such a feature set is based on the four categories previously enumerated, i.e. first considering capabilities, then operating environments, and finally domain technologies and implementation techniques.
2. Secondly, restrictions rules set up constraints and optimal selections, i.e. if one feature requires the existence of another one, the latter will be selected. On the other hand, two features may not be held concurrently; the selection of one of them rules out the other.

After that, those features not compiled in the domain model (e.g. security or performance features) are determined and included in the feature diagram.

Stage 3: System modeling. The system features diagram represents the most relevant system characteristics. However, such a model is not conceived to specify the system but to describe the learning requirements and instructional conditions. Furthermore, the last activity of the system analysis is to define a system model. Such a model comprises all the information required to specify both the learning context and the application. The model system will be broken down into four views or perspectives:

1. Structural view, which defines the properties and relationships between the entities represented in the business model. In our case, the structural view represents both the learning contents, and learners' characteristics.
2. Operational view, which represents the functional and administered operations of the system: (1) Supported activities - the set of operations to manage the system; (2) Learning services - instructional activities provided by the system. Such activities are divided into learning, monitoring and authoring.
3. Behavioral view, which is used to describe the flow of operations that are executed in order to successfully achieve the outcomes of the system. The behavioral view often also describes how the entities defined in the structural view are referenced as part of the operations.
4. Contextual view, which represents those conditions that determine the instructional process. The contextual view comprises information about the duration of the course, administrative issues, etc.

The final result is a description of the system (system-features diagram) and a specification of both the learning context and the application (structural, operational, behavioral and contextual views).

Learning and Teaching Programming

Having presented the principles of the analysis method, a practical application is next illustrated. The purpose of this case study is to show how this methodology operates and the usefulness of our solution. The *learning and teaching programming domain* has been selected, because:

- It is a well-known domain. There is a long list of references in which this domain is the object of interest.
- Developing programming courses is a complex process. The design of the course is mainly determined by the expertise of designers.
- Learning material reusability is a common but non-trivial outcome. Materials must be adapted to the characteristics of the students and the resources available.

Following the definition of the learning scenario, a set of activities needs to be carried out. First of all, as part of the 'domain analysis' process, the compilation of information on such a problem, the identification of the aspects that characterize the domain, and the classification of these aspects by considering the categories indicated above are completed. Thus, as an example, the following clauses describe a simplified version of the feature diagram shown in Figure 2:

- Capability. The characteristics included in this group refer to: (1) Profile of the student, which in the field of computer programming can be (Dreyfus & Dreyfus, 1986): novice, advance beginner, competence, proficiency and expert; (2) Contents, which according to Davies (1993), includes two kinds of content that can be learnt by a student, i.e. programming knowledge (e.g. loop sentence) and programming strategies (e.g. using a loop appropriately in a program); (3) Learning objectives, which can be grouped as follows: those regarding programming design, code implementation, and result evaluation (Robins, Rountree & Rountree, 2003).
- Domain technology. In general, the most adequate instruction design for learning programming is: reading, problem-based learning, and discovery learning. The latter can be carried out both individually and collaboratively (in groups of students).
- Implementation techniques for: (1) Activities, including computer room activities, guided laboratory and tutorials; and (2) Services, including collaboration, sequencing and group management.

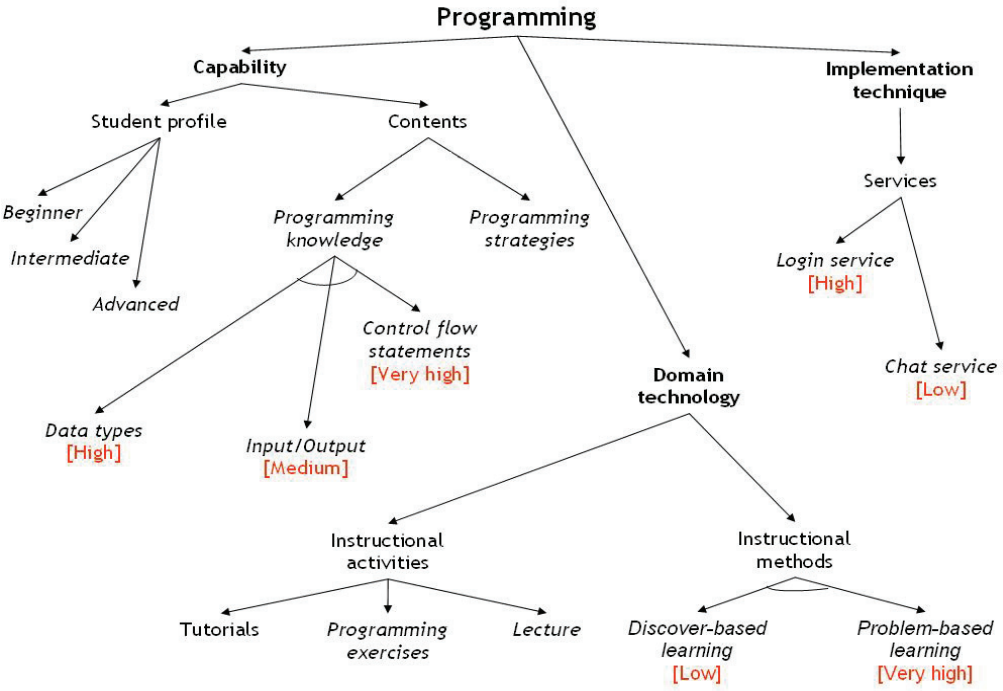

Figure 2: Domain-feature diagram. Learning and teaching programming domain.

Finally, a set of rules and conditions that relate these features and that establish the restrictions between them are defined. Examples of these restrictions are the following:

- The profiles of students are exclusive, i.e. a student cannot be allocated to two profiles at the same time.
- There is a dependency relation between contents, i.e. it is not possible to transmit contents on programming strategies without the students already having acquired prior knowledge on such contents.
- Problem-based learning is advisable so as to acquire knowledge on code implementation, and this requires carrying out laboratory-based activities.
- Group management services are not suitable for beginners. Novice programmers are very local and concrete in their comprehension of programming knowledge (Wiedenbeck, Ramalingam, Sarasamma & Corritore, 1999). Furthermore, they have problems defining appropriate algorithms and identifying the accurate strategy to resolve a specific and detailed problem (Robins et al., 2003). Because of these constraints, beginners have only limited skills to organize and manage their work.

The feature-domain model is useful as a reference for the analysis of diverse domain learning systems. At the preliminary stages of the development of a new system it is not obligatory to start from scratch, but it is possible to follow the guidelines of the feature model. Likewise, the process identifies the common elements among domain systems.

In the particular case of the *learning and teaching programming* domain, the 'system analysis' process for the design of a programming-related subject for the first course of the degree on Computer Engineering consists of the following steps:

- Determine the objectives of the course, the profile of the students, and the type of knowledge to be generated. Specifically, in our work, we wish to elaborate a programming course for novice students, focused on the transmission of knowledge that permits the coding of simple programs.
- Having validated possible restrictions to the capabilities selected, the instructional method to be used has to be identified. In this case, and after reviewing potential alternatives, it was decided to choose problem-based learning.
- Based upon the instructive method selected, and bearing in mind the restrictions of the model, the most appropriate activities were chosen (i.e. laboratory-based activities and guided laboratories)
- Finally, in order to complete the requirements identification stage, a revision of the features specific to the course, although not included in the domain model, was made, including specification of administrative characteristics (course duration, number of students, etc.), facilities and technological devices available, previous experience of students in on-line courses, and evaluation mechanisms.

The most innovative result of this analysis is the description of the system as a system-feature model. Thus, for instance, since one of the purposes of our course is to teach types of data, and given that this content is a common feature in the domain, it is likely to find learning objects that have already been produced. Taking into account this approach, the first stage of the analysis process contributes to having a better understanding of which elements the system design should concentrate on so as to enhance the reusability of learning objects.

Conclusion and Future Research Work

The application of software engineering in instructional design has turned out to be an appropriate working strategy to optimize the design and development of learning objects. According to this idea, an approach for the analysis of computer-supported learning systems has been presented. This approach, called Learning Analysis, proposes a systematic discovery and exploitation of commonalities across related learning domains for achieving successful reuse. By examining a family of related domain knowledge and the best practices underlying this knowledge, it is possible to obtain a set of reference models expressed in terms of 'features'. The model that captures the commonalities and differences is called a 'feature model', and it is used to support both engineering of reusable materials and the specification of the new learning system.

Furthermore, the use of a method based on domain analysis establishes the baseline for the definition of a complete method in Domain Engineering. In this case, the existence of groups of domain features would be associated or related to templates of learning material. As a result, it would be possible to automate the search and selection of the learning object to be reused.

Future work will lead to the refinement and specification of the guidelines here proposed, which constitute essential tools for the development of diverse models. The approach presented must be refined, elaborated and improved through its application to other real cases studies. Finally, the method will be assessed in two ways: its utility to analyze computer-supported learning systems, and its usefulness to identify reusable learning components.

The final objective will be the definition of a complete method of Domain Engineering that will allow transferring and adapting the guidelines defined for Product Line Engineering and Generative Programming (Dodero, Sánchez-Alonso & Frosch-Wilke, 2007) to the development of computer-supported learning systems.

References

Alessi, S. M., & Trollip, S. R. (1991). *Computer-based instruction: Methods and development*. New Jersey: Prentice Hall.

Alessi, S. M., & Trollip, S. R. (2001). *Multimedia for learning: Methods and development*. New Jersey: Allyn and Bacon.

Arango. G. (1994). Domain analysis methods. In W. Schafer, R. Prieto-Diaz, & M. Matsumoto (Eds.), *Software reusability* (pp. 17-49). London: Ellis Horwood.

Baggetun, R., Rusman, E., & Poggi, C. (2004). Design patterns for collaborative learning: From practice to theory and back. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2493-2498.

Clements, P., Northrop, L., & Northrop, L. M. (2003). *Software product lines: Practices and patterns*. Boston: Addison Wesley.

Cuadrado, J. J., & Sicilia, M. A. (2005). Learning object reusability metrics: Some ideas from software engineering. *Proceedings of the First International Conference on Internet Technologies and Applications.*

Czarnecki, K. (1998). *Generative Programming. Principles and techniques of software engineering based on automated configuration and fragment-based component models*. PhD Thesis, Technical University of Ilmenau.

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies*, *39*, 237–267.

Dick, W., & Cary, L. (1990). *The systematic design of instruction.* New York: Harper Collins.

Dijkstra, S., & Merriënboer, J. J. G. van. (1997). Plans, procedures, and theories to solve instructional design problems. In S. Dijkstra, N. Seel, F. Schott, & R. D. Tennyson (Eds.), *Instructional design: International perspective. Volume 2 - Solving instructional design problems* (pp. 23-43). New Jersey: Lawrence Erlbaum Associates.

Dodero, J. M., Sánchez-Alonso, S., & Frosch-Wilke, D. (2007). Generative instructional engineering of competence development programmes, *Journal of Universal Computer Science*, *13*(9), 1213-1233.

Dreyfus, H. & Dreyfus, S. (1986). *Mind over machine: The power of human intuition and expertise in the era of the computer*. Free Press.

Ferré, X., & Vegas, S. (1999). An evaluation of domain analysis methods. *Proceedings 4th CAiSE Workshop on Exploring Modelling Methods for Systems Analysis and Design*, *1999*.

Gagne, R. M., & Medsker, K. L. (1995). *The conditions of learning: Training applications*. Belmond: Wadsworth Publishing.

Goodyear, P. (1995). Infrastructure for courseware engineering. In R. D. Tennyson & A. E. Barron (Eds.), *Automating instructional design: Computer-based development and delivery tools* (pp.11-31). New York: Springer Verlag.

Goodyear, P. (1997). Instructional design environments: Methods and tools for the design of complex instructional systems. In S. Dijkstra, N. Seel, F. Schott, & R. D. Tennyson (Eds.), *Instructional design: International perspective. Volume 2 – Solving instructional design problems* (pp. 83-111). Kentucky: Lawrence Erlbaum Associates.

Grützner, I., Ruhe, G., & Pfahl, D. (2002). Systematic courseware development using an integrated engineering style method. *Proceedings of Networked Learning In A Global Environment: Challenges and Solutions for Virtual Education.* Retrieved November 21, 2007, from http://publica.fraunhofer.de/documents/N-9565.html

Hadjerrouit, S. (2007). Applying a system development approach to translate educational requirements into e-learning. *Interdisciplinary Journal of Knowledge and Learning Objects, 3*, 107-134. Retrieved from http://ijello.org/Volume3/IJKLOv3p107-134Hadj296.pdf

Hartshorne, R. (2003). Thoughtful creation of online course content: implications of SCORM for educators. *Academic Exchange Quarterly*.

Jones, R., & Boyle, T. (2007). Learning object patterns for programming. *Interdisciplinary Journal of Knowledge and Learning Objects, 3*, 19-28. Retrieved from http://ijello.org/Volume3/IJKLOv3p019-028Jones.pdf

Kang, K.C. (1998). FORM: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering, 5*, 143-168.

Koper, R. (2005). An introduction to learning design. In R. Koper & C. Tattersall (Eds), *Learning design. A Handbook on modelling and delivering networked education and training* (pp. 3–19). Berlin: Springer Heidelberg.

Lee, K., Kang, K. C., Chae, W., & Choi, B. W. (2000). Feature-based approach to object-oriented engineering of applications for reuse. *Software-Practice and Experience*, 1025-1046.

McGreal, R. (2004). Learning objects: A practical definition. *International Journal of Instructional Technology and Distance Learning, 9*(1). Retrieved November 21, 2007, from http://www.itdl.org/Journal/Sep_04/article02.htm

Padrón, C. L., Diaz, P., & Aedo, I. (2007). Towards an effective evaluation framework for IMS LD based didactic materials: Criteria and measures. *Proceedings of International Conference on Human Computer Interaction*, 312-321.

Paquette, G. (2004). *Instructional engineering in networked environments*. San Francisco: Pfeiffer.

Prieto-Diaz, R. (1990). Domain analysis: An introduction. *Software Engineering Notes, 15*(2), 47-54.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172.

Sicilia, M. A., & Garcia, E. (2003). On the concepts of usability and reusability of learning objects. *The International Review of Research in Open and Distance Learning*, *4(2).* Retrieved November 21, 2007, from http://www.irrodl.org/index.php/irrodl/article/view/155/702

Sicilia, M. A. (2004). Reusability and reuse of learning objects: Myths, realities and possibilities. *Proceedings of the First Pluri-Disciplinary Symposium on Design, Evaluation and Description of Reusable Learning Contents.* Retrieved November 21, 2007, from http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-117/paper38.pdf

Tennyson, R.D. (1995). Instructional system development: The fourth generation. In R. D. Tennyson & A. E. Barron (Eds.), *Automating instructional design: Computer-based development and delivery tools* (pp. 33-78). New York: Springer Verlag.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. L. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*, 255-282

Wiegers, K. E. (2003). *Software requirements: Practical techniques for gathering and managing requirements throughout the product development cycle*. Redmond: Microsoft Press

Biographies

David Díez holds a Computer Science Degree from the Universidad Autónoma de Madrid, and MSc in Computer Science and Technology from the Universidad Carlos III of Madrid. From 1998 to 2005, he worked as software engineering and project manager for different multinationals companies. Currently, he works as an assistant teacher at the Universidad Carlos III de Madrid, and he is doing his PhD Thesis concerned to technological support for learning systems.

Camino Fernández got her PhD at the Universidad Politécnica de Madrid, and works as Associate Professor at the Computer Science Dept. at the Universidad Carlos III de Madrid. Her main research interest is adaptive systems. She has applied ideas in various fields, ranging from robots to digital libraries or e-learning systems. She has published over 50 papers in journals, books, and conferences in these areas.

Juan Manuel Dodero holds a Computer Science Degree from the Universidad Politécnica de Madrid (1993) and a PhD in Computer Science from the Universidad Carlos III de Madrid (2002), where he is currently associate professor. He is co-author of more than a dozen publications on international journals and books, and more than 30 communications in international research conferences. In 2005, he received the IEEE TCLT young researcher award for his early post-doc research activities on learning technologies.