An Effective Analysis Method for Computer-Supported Learning Systems Reusability

David Díez, Camino Fernández, Juan Manuel Dodero

DEI Laboratory - Computer Science Department
Universidad Carlos III de Madrid, Spain
{david.diez, camino.fernandez, 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 optimise the learning material 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 materials. The purpose of this paper is to have an analysis method for learning systems that guarantees the quality of the development. This specific analysis method can facilitate the reusability and minimize the relevance of the expertise in the development of learning materials.

Keywords: Instructional engineering, computer-supported learning system, analysis method, reusability, commonality.

1 Introduction

Instructional Engineering is defined as a method that supports the production of computer-supported learning systems by integrating the concepts and principles of instructional design and software engineering [1]. The life-cycle of earlier instructional engineering methods resembles the phases of the traditional software engineering process: analysis, design, development and evaluation.

As Goodyear points out [2], the relation between software engineering and learning software 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."*

The analysis phase is an essential phase of the development process, as well as one of the determinants of product success [3,4]. These qualities are themselves very meaningful; nevertheless, we can identify another factor that makes the activity of analysis one of the most interesting activities in the development of computer-supported learning systems: effective reuse must be planned and considered early in the life cycle [5].

Reuse is one of the most important issues in learning systems development. So far, there have been many attempts to support product reuse, but most of these efforts have focused on defining reusable patterns [6] and on designing artifacts for evaluating and recovering materials [7]. Therefore, according to the experience in software development, efforts should be put in defining methods for discovering commonalities in early stages, and using this information to achieve reuse.

The rest of the paper is organized as follows. Section 2 describes the motivations for a specific analysis approach in the development of computer-supported learning systems. The features of our analysis method are described in section 3. Section 4 includes a case study on how the approach has been applied. Finally, section 5 compiles a set of conclusions and future work.

2 The motivation for a specific analysis method

Courseware is defined as any instructional system delivering content or assignments via computers that supports learners as well as teachers in their educational efforts, in all technical and instructional way [8]. The life-cycle of the earlier courseware development methods resembles the phases of the traditional software development process: analysis, design, development and evaluation. Taking the opinion of diverse authors as a starting point that grants to the analysis phase a preponderant role [9], 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.

2.1 The function of the analysis stage

The analysis is the front end phase of the development process of computer-supported learning systems. In spite of its relevance, there is not an homogeneous view on the usefulness or functionality of the analysis stage in the development process:

– The methods derived from instructional design 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. The MISA method [1], the Alessi & Trollip method [10] and the Dean & Whitlock [11] are included within this category.
– The development models based on principles from software engineering propose an analysis phase in which the user needs are described. In this case, the analysis aims at studying the system requirements, focusing on contents and the presentation style. E.g.: models based on prototype techniques [12].

The close relation between software engineering and instructional design suggests that the analysis for the development of computer-supported learning systems should encompass principles from software development and instructional system development. Nevertheless, as indicated previously, this idea is not wholly attested.

2.2 The relevance of the reuse

In the field of the development of computer-supported learning systems, reusability refers to the ability to use content or learning objects for multiple courses or lessons [13]. Reusability is considered as an essential and arguably the most important characteristic of learning objects [14]. However, since reusability refers to prospective and future usage scenarios, it is difficult to achieve.

The concept of reusability encompasses aspects related to format, interpretation and pedagogical suitability (15). Thus, in order to achieve an adequate reusability it is necessary to tackle the problem from the point of view of:

– The learning object. The correct definition and formalization of learning objects allows determining their potential reuse for specific contexts.
– The learning contexts. The adequate specification of the learning context permits determining the most appropriate instructive method and, therefore, identifying the more susceptible tasks or activities to reusability.

So far, the standards and tools available have focused on the first of the views presented above by dealing with aspects regarding format (but which, on the other hand, overlooks the analysis of the specific context). In an attempt to overcome this matter, the study of reusability in the development process of computer-supported learning systems has been proposed.

Based on the experience of software development, reusability should be considered at the early phases of the development process [5]. For this reason, and in order to facilitate the identification of commonalities and further reuse, the definition of systematic analysis methods is needed.

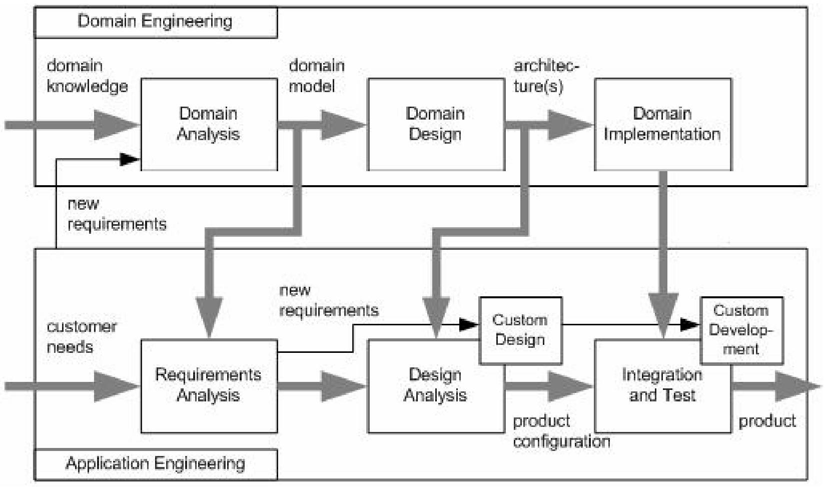

Fig 1. The software development process based on domains.

3 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 will allow defining a complete approach of instructional engineering analysis. Learning Analysis is conceived as a systematic instructional engineering analysis method that enables us to improve the reusability.

3.1 Engineering principles

Learning Analysis is based on Product Line Engineering [16] and, specifically, on Domain Engineering [17]. Domain Engineering is an activity that helps to build reusable components [17]. As shown in figure 1, Domain Engineering addresses the systematic creation of domain models, which represent the set of requirements that are common to systems within a product line.

Several software engineering approaches propose a careful analysis of the domain; however, feature modelling is the most effective technique to represent the domain knowledge [5]. The underlying motivations for using feature modelling to represent the learning domain are:

- Users 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 [19].
- The reusable software contains inherently more variability than specific applications, and feature modelling is the key technique for identifying and capturing commonality and variability [18].

The feature modelling captures the general capabilities of an application in a domain in terms of *features*. A feature is defined as [18]:

> *"A distinguishable characteristic of a system concept that is relevant to some stakeholder in a specific domain."*

A feature model represents the common and the variable features of the concepts involved in the problem and the dependencies between the variable features. A feature model consists of a feature diagram and some additional information, such as a short semantic description of each feature. Different types of features concern different types of interests in systems development. Kang [5] indicates that application features can be classified into four different categories: (1) *Capability features*, which are associated with the services or functions provided by the system and which constitute the main concern of users; (2) *Domain technology features* and (3) *operating environment features,* which are the concern of designers; and (4) *Implementation technique features, which* are the concern of developers. Applications cannot be built unless these different groups of features are decided upon by each respective group of experts.

In our approach these groups have been adapted to the educational field by defining the following categories:

- Capability. It corresponds to contents, learning objectives and the attributes of potential students.
- Domain technology: Platforms, tools and instructional standards.
- Operation environment. It compiles the instructional theories and the most suitable methods to the domain.
- Implementation technique. Learning activities and services used in the domain.

3.2 Engineering process

In addition to the engineering principles, Learning Analysis provides a well-defined engineering method. The analysis method is broken down into two processes:

- Domain analysis. Activities for analysing the domain knowledge and creating a model. The process consists of three activities:
  - Context analysis. The activity defines the scope of the domain. In addition, it compiles information on the domain. The information can be provided by experts or collected from available references and systems.
  - Feature identification. Once the context is scoped, the domain modelling phase provides the steps to identify the features of the domain. This activity receives information from the following sources: rules derived from instructional design theory, rules derived from best practices, and rules derived from patterns in best practices [20].
  - Domain modelling. Having identified the features, they need to be classified and organised in accordance to their nature. Such a classification is based on the categories enumerated previously. In addition to the feature model, a set of rules and restrictions amongst features is defined.
- Application analysis. Activities for analysing a specific learning system by using the domain model created in the domain analysis stage. The process aims at identifying the features of a specific computer-based learning system. The selection of features is done by following some guidelines and by taking as a reference the domain model. An effective method for identifying such a feature set is by following the four categories, i.e. first considering capabilities, then operating environments, and finally domain technologies and implementation techniques. Finally, those features not compiled in the domain model are determined.

The use of an analysis method divided into two independent processes, although interrelated, 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 variabilities of systems within a domain [21]. The results of the domain analysis are used by the application analysis to identify potential elements for reuse in a specific application.

4 Case Study: Learning and Teaching Programming

Having presented the principles of the analysis method, a practical application is illustrated. The purpose of this case study is to show how this methodology operates and the usefulness of the solution proposed.

The first step will be to determine the working context. With this goal, a base definition is proposed and taken as a reference to set the scope of the domain. For the purpose of our work, a domain is defined as:

> *"A set of educational contexts that share certain contents, as well as pedagogical objectives and a specific form of instruction. The magnitude of the educational contents considered must be such that they are meaningful by themselves"*.

As an example of domain, and a case study, the *learning and teaching programming domain* has been selected. It is a very interesting learning domain 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 [15].
– Learning material reusability is a common activity.
– Reusability is a non-trivial task. Materials must be adapted to the characteristics of the students, the instructive method and the resources available.

Following the definition of the problem, a set of activities needs to be carried out: compilation of information on such a problem, identification of the aspects that characterise the domain, and classification of these aspects by considering the categories indicated above. Thus, as an example, the following paragraphs describe a simplified version of a feature model:

– Capability. The characteristics included in this group refer to: 1) Profile of the student, which in the field of computer programming can be [22]: novice, advance beginner, competence, proficiency and expert. 2) Contents. According to Davies [23], there are two kinds of contents that can be learnt by a student: programming knowledge (e.g. loop sentence) and programming strategies (e.g. using a loop appropriately in a program). 3) Learning objectives. Learning objectives can be grouped as follows [24]: those regarding programming design, code implementation and result evaluation.
– Operation environment. In general, the most adequate instruction design for programming learning is: reading, problem-based learning and discovery learning. The latter can be carried out both individually and collectively (groups of students).
– Implementation technique. 1) Activities, including laboratory activities, guided laboratory and tutorials. 2) Services, including collaboration, sequencing and group management.

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

1. The profiles of students are exclusive: the student cannot be allocated to two profiles at the same time.
2. There is a dependency relation between contents: it is not possible to transmit contents on programming strategies without acquiring prior knowledge on such contents.
3. Learning objectives are independent. It is not compulsory to have knowledge on a particular group in order to gain knowledge on other groups.
4. Problem-based learning is advisable so as to acquire knowledge on code implementation.
5. Problem-based learning requires carrying out laboratory-based activities, either guided or not-guided.
6. Group management services are not suitable for beginners.

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, by means of the simplest and most effective way, the common elements among domain systems. In the particular case of the *learning and teaching programming* domain, the analysis process for the design of a programming-related subject for the first course of the degree on Computer Engineering would consist of the following steps:

– Firstly, to 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 programmes.
– Having validated possible restrictions to the capabilities selected, the instructive method to be used has to be identified. In particular, and after reviewing potential alternatives, it was decided to opt for problem-based learning.
– Based upon the instructive method selected, and bearing in mind the restrictions of the model, the most desirable activities were chosen (i.e. laboratory-based activities and guided laboratories).
– As a final step, and in order to complete the analysis stage, a revision of the features specific to the course, although not included in the domain model, was made; amongst others: 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 final result of the analysis phase is a model of the specific features of the course to be developed, which will include the common aspects (compiled in the domain model) and the particularities of the learning system under design. As shown in the example, the use of a knowledge model that takes into consideration its features eases the reuse of learning objects in the following two ways:

– On the one hand, it identifies common elements of the course in the context: those compiled in the feature model of the domain. Thus, it is straightforward to predict those elements susceptible to reuse (for which some material might have already been developed).
– On the other hand, reusable aspects are characterized, what facilitates the search and selection of available material.

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 automatise the search and selection of the learning object to be reused.

5 Conclusion

The application of software engineering to instructive design has turned to be an appropriate working strategy to optimise the development of computer-supported learning systems. According to this idea, an approach for the analysis of instructive software has been presented. This approach, called Learning Analysis, aims at improving the development process in order to:

– Facilitate the reuse of materials.
– Minimize the relevance of experience in the development of learning materials.

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 “feature model”, and it is used to support both engineering of reusable materials and the specification of the new learning system.

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, stretched and improved through its application to others real cases of study. 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 to the development of computer-supported learning systems.

Acknowledgements. This work is supported by the Moduweb project (TIN2006-09678) funded by the Spanish Ministry of Science and Education (MEC)

References

1. Paquette, G. Instructional Engineering in Networked Environments. Pfeiffer, San Francisco. (2004)
2. Goodyear, P. Infrastructure for courseware engineering. In Tennyson, R.D. and Barron, A.E. (eds.): Automating Instructional Design: Computer-Based Development and Delivery Tools. Springer Verlag, New York (1995) 11-31
3. Gagne, R.M. and Medsker, K.L. The Conditions of Learning: Training Applications. Wadsworth Publishing, Belmond, California (1995)
4. Preece, J., Rogers, Y. and Sharp, H. Interaction Design: beyond human computer interaction. John Willey & Sons, New York (2002)
5. Kang, K.C. FORM: A feature-oriented reuse method with domain-specific reference architectures. Annals of Software Engineering 5 (1998) 143-168
6. Baggetun, R., Rusman, E. and Poggi, C. Design Patterns for Collaborative Learning: From Practice to Theory and Back. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (2004) 2493-2498
7. Padrón, C.L., Diaz, P. and Aedo, I. On the Concepts of Usability and Reusability of Learning Objects. The International Review of Research in Open and Distance Learning (2003) Vol. 4, No. 2
8. Grützner, I., Ruhe, G. and Pfahl, D. Systematic Courseware Development Using an Integrated Engineering Style Method. In Proceedings of NETWORKED LEARNING IN A GLOBAL ENVIRONMENT: Challenges and Solutions for Virtual Education (2002)
9. Wiegers, K. E. Software Requirements: practical techniques for gathering and managing requirements throughout the product development cycle. Microsoft Press, Redmond. (2003)
10. Alessi, S.M. and Trollip, S.R. Computer-Based Instruction: Methods and Development. Prentice Hall, Nueva Jersey (1991)
11. Dick, W. and Cary, L. The Systematic Design of Instruction. Harper Collins, Nueva York (1990)
12. Goodyear, P. Instructional design environments: Methods and tools for the design of complex instructional systems. In Dijkstra, S., Seel, N., Schott, F. and Tennyson, R.D. (eds.): Instructional Design: International Perspective. Volume 2 – Solving Instructional Design Problems. Lawrence Erlbaum Associates. Kentucky (1997) 83-111
13. Hartshorne, R. Thoughtful creation of online course content: implications of SCORM for educators. Academic Exchange Quarterly (2003)
14. Sicilia, M.A. and Garcia, E. On the Concepts of Usability and Reusability of Learning Objects. The International Review of Research in Open and Distance Learning (2003) Vol. 4, No. 2
15. Sicilia, M.A. Reusability and reuse of learning objects: myths, realities and possibilities. In Proceedings of the First Pluri-Disciplinary Symposium on Design, Evaluation and Description of Reusable Learning Contents (2004)
16. Clements, P., Northrop, L. and Northrop, L.M. Software Product Lines: Practices and Patterns. Addison Wesley, Boston (2003)
17. Prieto-Diaz, R. Domain Analysis: An Introduction. Software Engineering Notes (1990) Vol. 15, No. 2, 47-54

18. Czarnecki, K. Generative Programming. Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models. PhD Thesis, Technical University of Ilmenau (1998)
19. Lee, K., Kang, K.C., Chae, W. and Choi, B.W. Feature-based approach to object-oriented engineering of applications for reuse. Software-Practice and Experience (2000) 1025-1046
20. Koper, R. An Introduction to Learning Design. In. Koper, R. and Tattersall. C. (eds): Learning Design. A Handbook on Modelling and Delivering Networked Education and Training, Springer Berlin Heidelberg (2005) 3–19
21. Prieto-Diaz, R. Domain Analysis: An Introduction. Software Engineering Notes (1990) Vol. 15, No. 2, 47-54
22. Dreyfus, H. and Dreyfus, S. Mind over machine: The power of human intuition and expertise in the era of the computer. Free Press (1986)
23. Davies, S.P. Models and theories of programming strategy. International Journal of Man-Machine Studies, 39 (1993) 237–267
24. Robins, A., Rountree, J. and Rountree, N. Learning and Teaching Programming: A Review and Discussion. Computer Science Education (2003) Vol. 13, No. 2, 137-172