A VISUAL METHODOLOGY FOR THE DESIGN OF COURSE CONTENTS
A Case Study in Communications Software

Jesús Martínez
Computer Science Departament, University of Málaga, Spain

Keywords: Content development, UML, Model-driven architecture.

Abstract: This paper proposes a novel methodology for the design of courses visually. Our method takes borrowed techniques from Software Engineering such as the Unified Modeling Language (UML) and Model-driven Architecture (MDA). The main goal of this approach is the creation of generic courses that may be refined later depending on different educational contexts. Therefore, teachers will have automatic support from tools in order to schedule lectures, activities or resources. This experience is being carried out in the context of a *Communications Software* course for different undergraduate degree programs at the University of Málaga.

1 INTRODUCTION

During the last thirty years, distributed services have evolved in the same way as computer networks and protocols have. Part of the success of these newly emerging applications is due to the use of high-level standards and middleware which allow developers to abstract specific features of transport protocols and execution platforms. However, the availability of these resources may lead to the design of poor performance communications software (due to the abuse of application protocols and Web Services, misunderstanding of the Socket API,...). It is worth noting that many networked applications are thought to be efficient only because less time was spent on their development, thus some important details regarding robustness and scalability may not have been considered. Our students must understand that in order to design good communications software they will have to obtain a good founding in Concurrent Programming, Protocol Engineering and Software Engineering. For this reason, it is important that institutions and teaching staff must carefully consider what, when, and how to teach these subjects.

For instance, a Communications Software syllabus usually focuses on different learning objectives if it belongs to different programs (Computer Engineering, Electrical Engineering), degrees (undergraduate, masters of science) and also depends on the number of hours allocated to the course. Moreover, there are other factors that influence coursework planning, for instance, if the students skills on programming languages are not as expected or if they lack an in-depth knowledge of operating systems or concurrency. In these cases, there will be more introductory lectures, usually at the expense of other material later due to lack of time. Therefore, the original syllabus becomes obsolete. This situation applies to some of these courses in the Electrical Engineering and Computer Engineering programs existing in the University of Malaga where students may have taken different courses previously: rigid planning significantly affects the academic performance of some students.

Designing a course taking all these constraints into consideration would be a challenge for any teacher so it would be more beneficial to handle the learning objectives, lectures and activities at different levels of detail (from more general to more specific ones), where each level would deal with specific requirements according to the constraints considered. Undoubtedly, in order to obtain a model for designing courses in this way, we should apply a rigorous methodology to support the description, manipulation and refinement of each level, until the most suitable course is obtained.

This paper presents a visual approach for the design of contents and learning activities for a course. We will use Software Engineering methods and tools such as the Unified Modeling Language (UML) (Object Management Group, c) to describe subjects, con-

Martínez J. (2010).
A VISUAL METHODOLOGY FOR THE DESIGN OF COURSE CONTENTS - A Case Study in Communications Software.
In *Proceedings of the 2nd International Conference on Computer Supported Education*, pages 434-437
DOI: 10.5220/0002792204340437

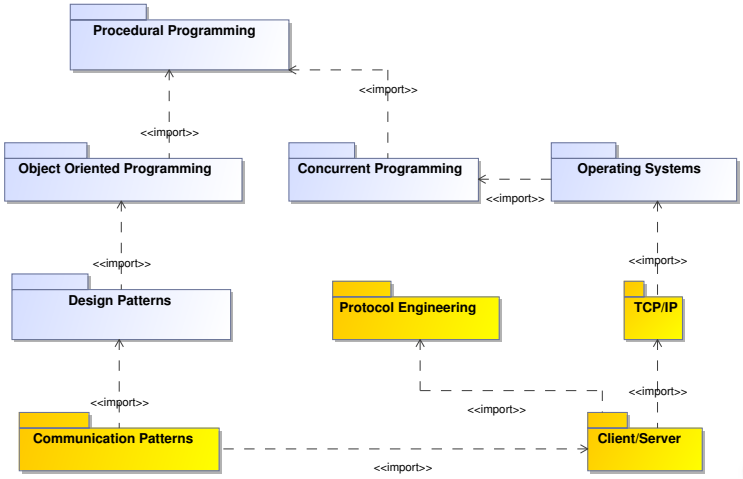

Figure 1: A visual diagram describing relationships among subjects.

cepts and their relationships visually. We will also use UML to define activities and their sequencing during the course. It is worth noting that there are only a few approaches which have employed UML to describe some concepts related to education (Nodenot et al., 2004; Marlowe et al., 2005). We also propose to design courses in a generic way, that is, we will use some guidelines available in the Model-driven Architecture (MDA) (Object Management Group, a) to transform a generic (and visual) UML course in order to obtain specific subjects according to different educational contexts or the student backgrounds. MDA transformation rules will also be useful to obtain other resources from our visual courses, such as documentation for teachers, subject coordinators or students (the syllabus, mandatory exercises, etc.). Courses designed using this approach are stored in XML format, it being possible to manipulate them automatically using well-known CASE tools, such as Eclipse (The Eclipse Foundation, 2009).

2 APPLYING UML AND MDA FOR DESIGNING COURSES

Communications Software and related subjects are part of the Computer and Electrical Engineering curricula available in the University of Málaga. This section presents a case study for the design of *Communications Software* related courses using UML and MDA concepts in two steps. First, we will define subjects, their concepts and relationships in a visual (and generic) way. We will also describe a set of activities to be carried out (sequentially or in parallel) within every subject. Second, we will define the appropriate transformation rules to obtain a specific curriculum (for concepts and activities), taking into account different educational contexts or the student backgrounds.

2.1 Designing Generic Courses

In order to use our visual approach, subjects in the curriculum are first selected following the University guidelines. Once we have the set of learning objectives to cover, we will design generic subjects, the concepts to cover and activities. Initially, we will depict high-level Class Diagrams to model these subjects which will include contents units. We will also indicate relationships between subjects (including those belonging to other existing curricula in the program). Fig. 1 shows an example of this description, where each subject constitutes a UML package. In the figure, available packages refer to Socket TCP/IP programming, Client/Server architecture or specific design patterns for communications (Schmidt and Huston, 2003) contents. The subject called Protocol Engineering is based on teaching typical concepts needed to build protocol stacks in layers: messages, queues, timers or state machines, among others. The relationship between subjects (or content units) will be explicitly depicted in the diagram, beginning with the basis set by Procedural Programming, taught in the first courses. An `<<import>>` relation indicates that one subject requires the previous knowledge obtained in another one. This dependence relationship is clarified inside each subject/package using a Class Diagram where topics or concepts will be described as UML classes, allowing the connections with previous concepts through class relationships. Fig. 2 depicts a simplified scenario where some classes de-

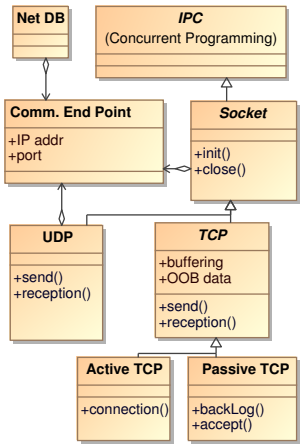

Figure 2: A generic model to describe Socket concepts.

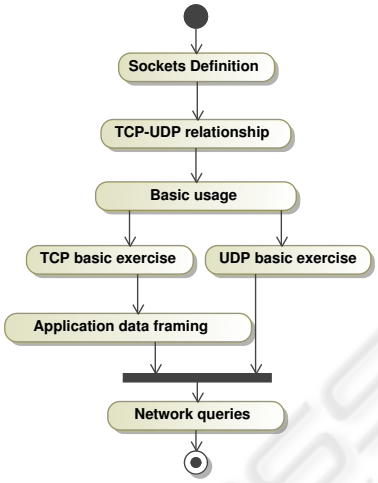

Figure 3: Sequence of activities involved in a subject.

fine concepts of the TCP/IP subject. The figure shows the concept of *Socket*, which is derived from a previous concept called *IPC* (inter-process communication concept), which was learned previously within the Concurrent Programming subject. The *Socket* concept will be specialized in **UDP** and ***TCP*** sockets. For each class, the attributes are used to list the main important features of the concepts. For example, when defining a **Communication End Point**, one IP address and one transport port are required. Procedural concepts are included as operations within the class. We have included the basic operations that students will use when working with Sockets. The diagram in fig. 2 also uses more UML notation for our description purposes. For instance, classes in italics (UML abstract classes) will now mean that the concepts they represent need to be extended/refined by other classes (that is, they need more explanation). This refinement is shown in fig. 2, where the IPC concept is specialized as a Socket and, then as a TCP socket, which is also specialized in a passive and active one. As commented before, this example describing a subject can be considered a generic model, and can be refined later.

2.2 Scheduling Lectures and Activities

Once we have designed our subjects and concepts using Class Diagrams, the next step in our methodology consists of organizing all activities which lead to obtaining the proposed learning objectives. Thus, we will add UML Activity Diagrams to our general model. Fig. 3 shows an example of a diagram prepared to be used for teaching the concepts related to the TCP/IP subject. An action could be fully identified by its name. Furthermore, control flow makes it easy to move forward through the syllabus. For example, if we are now in the action called *Basic Usage*, then we will execute two exercises in parallel, for example, one exercise programming TCP sockets (in the lab) and another one programming UDP sockets (at home).

2.3 Obtaining Specific Courses

In our methodology, an educational itinerary will be the result of a (parameterizable) transformation process applied to a generic course in order to obtain another more concrete. Therefore, the teacher could be able to react to different constraints such as the student backgrounds, and subsequently preparing different learning strategies (modifications of the generic elements designed in our visual subject). This transformation process will mean the existence of an MDA transformation template, which will be applied to our existing general models. The model obtained after the transformation will be a specific one, now including modifications in the original diagrams. A typical scenario of application for an educational itinerary, would consists of insertions, modifications and removals of actions in existing Activity Diagrams, or content features and explanations in Class Diagrams. In the proposed *Communications Software* course, we have considered at least three different itineraries: one that uses the C language and a UNIX platform, one that uses multiplatform C++ libraries and specific Design Patterns (Schmidt and Huston, 2003) and the last one using the Java language. For instance, fig. 4 shows a partial view of applying the so-called C/UNIX educational itinerary to the Class Diagram for TCP/IP concepts of fig. 2. In the figure, basic concepts on the left part become the ones on the right, which specify that the concepts will focus on

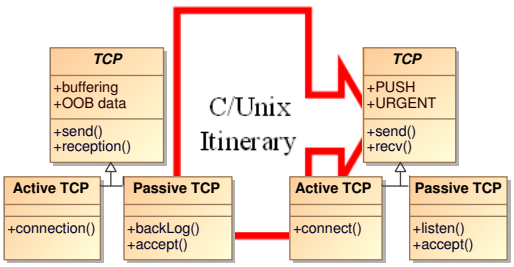

Figure 4: Obtaining a specific learning context.

Figure 5: Using a profile with our elements.

Berkeley Sockets for UNIX systems (and its common interface with send(), recv(), connect(), listen(), accept(), and other system calls). In the same way, these itineraries could be used to modify basic activities for concrete actions (lectures, practical exercises oriented to programming languages and platforms) along with the whole course planning.

2.4 Supporting Tools

In order to help automatic tools to detect and manipulate our proposed visual elements (subjects, concepts, activities,...), we are developing a UML Profile. This UML mechanism defines a so-called Domain Specific Language (DSL), giving the possibility to redefine the semantics of any UML elements as our approach requires. Therefore, UML elements redefined using a Profile will include specific *stereotype* tags to mark them and guide other processing tools such as the transformation engines. The most suitable stereotypes for our approach are being tested at this moment. Fig. 5 show an example of some stereotypes proposed. The left column depicts a special package with the `<<subject>>` semantics that could contain some `<<concept>>` classes, whereas the right column depicts two possible activities including a lecture and homework. These activities also include tags related to their duration (in minutes) and marking (valid for the homework). It is worth noting that the use of profiles has also been used in other e-Learning contexts, as presented in (Nodenot et al., 2004).

Therefore, a tool which supports our approach will have to include graphical interfaces to load, edit and save visual courses (stored in XMI format (Object Management Group, b)) and will have also to include features to create educational itineraries, where target elements to transform could be specified easily. These itineraries should be described visually in order to hide the complexities of a transformation language from teachers. Currently, we are implementing this graphical tool in Eclipse.

2.5 Conclusions and Future Work

This paper has presented a visual methodology for the design of courses and curricula using UML and MDA. Our aim is also to facilitate the coordination tasks and to detect inconsistencies and overloads of work on student activities during the courses.

ACKNOWLEDGEMENTS

The author would like to thank ETSI Informática and the University of Málaga for their support during this experience.

REFERENCES

Marlowe, T., Ku, C., and Benham, J. (2005). Acm sigcse bulletin, vol-37(1). In *Design patterns for database pedagogy: a proposal*.

Nodenot, T., Marquesuzaa, C., Laforcade, P., and Sallaberry, C. (2004). Model based engineering of learning situations for adaptive web based educational systems. In *Proc. 13th WWW Alternate*.

Object Management Group. MDA Guide Version 1.0.1.

Object Management Group. MOF 2.0/XMI Mapping, v2.1.1.

Object Management Group. UML Version 2.1.2.

Schmidt, D. and Huston, S. (2003). *C++ Network Programming Volume 2: Systematic Reuse with ACE and Frameworks*. Addison-Wesley.

The Eclipse Foundation (2009). Eclipse IDE. Available at http://www.eclipse.org/.