Designing a Pedagogical Model for Web Engineering Education: An Evolutionary Perspective

Said Hadjerrouit
Agder University College, Kristiansand, Norway

Said.Hadjerrouit@hia.no

Executive Summary

In contrast to software engineering, which relies on relatively well established development approaches, there is a lack of a proven methodology that guides Web engineers in building reliable and effective Web-based systems. Currently, Web engineering lacks process models, architectures, suitable techniques and methods, quality assurance, and a systematic approach to the development process. As a result, Web engineering is still struggling to establish itself as a reliable engineering discipline. The cost of poor reliability and effectiveness has serious consequences for the acceptability of the systems. One of the main reasons for the low acceptance of Web-based applications is the large gap between design models and the implementation model of the Web. It is therefore not surprising that Web engineering education still focuses on technologies, rather than on critical skills that facilitate engineers to solve real-world problems effectively. To meet the challenges of Web engineering, current education must be aligned with a pedagogical model capable of empowering and supporting the acquisition of critical skills. To do this, a new learning environment must be created that promotes change in both pedagogy and course material, in effect, altering the role of the teacher, the expectations for students, and many other educational aspects. This article describes a learner-centered pedagogical model rooted in the constructivist learning theory for teaching students the skills they need to construct Web-based applications more effectively and collaboratively. The model is iterative and encompasses two types of components: the learning management process and learning phases. The learning management process is concerned with monitoring learning activities, such as providing information, assessing project work, etc. The learning phases are an orderly set of interdependent activities moving from context analysis to the communication of learning results. The approach described in this paper is a result of 5 years of teaching Web engineering. The process of designing a pedagogical model can be characterized as an evolutionary process that progresses through a series of experimentations, evaluations, and redesigns. In evaluating the approach after many years of experience, the instructor can draw the following conclusions. First, applying the constructivist learning theory is clearly a challenge for instructors as it requires making a radical shift in their thinking in order to translate the philosophy of constructivism into practice. Second, it takes time and considerable effort to make significant pedagogical changes. Yet constructivism holds important lessons for how to design environments to support active learning. It gives teachers a framework for understanding stu-

Editor: Eli Cohen

dents' needs and motivations. It helps teachers to expose students to many aspects of the subject matter that are of crucial importance for the profession of Web engineering. It allows focus on what really matters for students – the acquisition of critical skills, authentic tasks, motivational aspects, teamwork, collaboration and negotiation, reading and writing skills, formative assessment, and self-evaluation.

Keywords: Constructivism, Learning Environments, Objectivism, Object-Oriented Modeling, Pedagogical Model, Unified Modeling Language, Web Engineering, Web Technology.

Introduction

Since the first workshop on Web Engineering in 1998, the construction of Web applications has made a lot of improvements, but there is still a lack of a disciplined engineering approach for building Web-based systems. According to the Web Engineering Community Portal (http://www.webengineering.org/default.aspx), the Web

> "has gone far beyond presenting information of research groups as in its early beginning. Although the applications become increasingly complex, the development process still remains ad-hoc. A large gap between design models and the implementation model of the Web has been recognized to be one of the main reasons for the low acceptance of disciplined development in the Web. It becomes clear that the construction and evolution of applications for the World Wide Web requires support such as is available for traditional applications through process models, architectures, methods and principles of software engineering. This new discipline that has been established during the previous years, is therefore seen as: Web Engineering - The application of systematic, disciplined and quantifiable approaches to the cost-effective development and evolution of high-quality applications in the World Wide Web."

However, even if Web engineering requires support from traditional applications, it has a broader context than traditional software engineering. It is a combination of software engineering, hypermedia and multimedia engineering, marketing, graphic design, cognitive science, and human-computer interaction (Balasubramaniam, Pries-Heje, & Baskerville, 2003; Escalona & Koch, 2004; Murugesan, Deshpande, & Hansen, 1999a, 1999b; Murugesan & Ginige, 2001; Whitehead, 2004). Hence, Web engineering needs a multidisciplinary approach that must rely on knowledge and expertise from different disciplines, such as software engineering, hypermedia engineering, human-computer interaction, cognitive science, marketing, ethics, etc. Thus, Web engineering encompasses inputs from diverse areas that can be divided into three categories: (a) Technological dimensions; (b) Engineering dimensions; and (c) Esthetical, ethical, legal, cognitive, organizational, and cultural dimensions.

First, in contrast to traditional software systems that are built using an homogeneous technology, Web-based applications run in a heterogeneous computing environment that includes multi-platforms, ulti-browsers, and multimedia support. This environment has Web-based programming languages and technologies, such as Java, JavaScript, HTML, CGI Script, PHP; Web servers and databases, Web editors, such as Microsoft FrontPage and Micromedia Dreamweaver, and many other means of implementation that provide support for the development of Web-based applications.

Second, the construction of Web-based systems is affected by engineering issues, e.g. process models, such as waterfall, evolutionary prototyping, spiral model or a combination of them; object-oriented approaches; hypermedia and multimedia engineering, usability engineering, Web design guidelines, software reuse, design patterns, architectural frameworks, and modeling languages (Bleek, Jeenicke, & Klischewski, 2004; Conallen, 1999; Hadjerrouit, 2001; Lowe & Eklund, 2002; Pressmann, 2001).

Finally, Web Engineering is affected by marketing, cognitive, ethical, legal, and cultural issues (Murugesan & Ginige, 2001). Many Web-based systems are indeed marketing channels both for private and public organizations. Hence, it is important to consider how to present the information that supports marketing. Then, cognitive issues play a central role in Web applications due to the cognitive effort required for users to manage the information space in the Web. In addition, Web-based systems are more user-oriented than traditional software systems. Thus, a significant part of any Web application concerns esthetical issues to produce look and feel of Web pages. Web engineering is also affected by legal constraints, ethical conventions, security against manipulation, copyright, disability discrimination, etc. Finally, many Web applications transcend the national boundaries. They become much more widespread in their use than traditional software systems. Thus, engineering for the Web should relate to diverse cultural contexts.

Web Engineering Skills

Given the state of Web engineering, the most influential factor for the success of Web engineering projects is the skill level required from Web engineers to master the development process. Thus, an important step in teaching Web engineering consists of identifying the skills that one expects Web engineers to possess (Reichgelt et al., 2004). According to (Seffah & Grogono 2002), three groups of skills are required for any software engineer. Likewise, Web engineers are expected to possess three groups of skills (Table 1):

1. The first group, prerequisite skills, is required for any student entering the field of Web engineering. The skills must be acquired at an early stage, because the learning of Web engineering depends on.
2. The second group is concerned with specific Web engineering skills. It provides the ability to perform key tasks of the Web engineering development process.
3. The third group, generic skills, is concerned with writing, reading, communication, dialogue, teamwork, and project planning. These skills are essential for Web engineers in a work situation.

Table 1: Web Engineering Skills.

Prerequisite skills:

1) Object-oriented modeling and programming with UML and Java, or similar languages.
2) Database development with the JDBC, MySQL and Java Servlets, or similar languages.
3) Web programming with HTML, JavaScript, CGI Script, and PHP, or similar languages.
4) Deployment of specialized authoring tools and Web editors, such as FrontPage and Macromedia Dreamweaver for the design and implementation of Web-based applications.

Specific skills:

1) Understanding Web engineering as a multidisciplinary field incorporating technical, engineering, social, political, marketing, legal, ethical, cultural, esthetical, and pedagogical issues.
2) Understanding the Web engineering development philosophy.
3) Understanding the context and system scope of Web engineering applications.
4) Analysis modeling: understanding the problem requirements; specifying users' requirements using use cases and scenarios; specifying data requirements using classes and other analysis modeling techniques; and general quality attributes for Web-based applications.
5) Design modeling: architecture design; component, logical, and deployment design; cohesion and coupling issues; elaboration of requirements and design criteria.
6) Web design: site design using linear, grid, hierarchical, and network Web structures; page design; interface design; navigation design, visual design; design of typography, editorial style, screen, colors, graphics, and multimedia.
7) Prototyping, incremental, evolutionary, and iterative development; program coding, unit and integration testing, evaluating and debugging of the evolving code solution, and consistency checking.
8) Usability testing and engineering; human-computer interaction; usability criteria in terms of user satisfaction, ease-of-use; ease-of-learn, and consistency.
9) Reflecting, evaluating, explaining, and justifying Web engineering solutions.

Generic skills:

1) Project planning and management.
2) Reuse of design principles, frameworks, architectures, and toolkits (class libraries); modifying and reusing existing analysis, design and program code solutions and patterns; comparing, contrasting, recognizing similarities and differences between new problems and previous solutions.
3) Writing and reading skills: writing and formatting technical documentation and reports, reading texts and documents.
4) Dialogue and communication with stakeholders. Working in teams with developers, system designers, programmers, end-users, and clients.

Pedagogical Foundation

Teaching a subject matter, which is still evolving, as it is the case of Web engineering, is both an exciting and challenging task. This paper illustrates how the learning process was designed to teach the skills one expects from Web engineers. Important for designing the learning process are learning theories. Literature reviews suggest that theories can be related to two main commonly accepted paradigms: The objectivist and the constructivist paradigm.

In terms of instruction, objectivism assumes that the goal of learning is to efficiently transmit knowledge from the instructor to the learners. Its metaphor of mind is that of a blank page, waiting to be filled with knowledge. Instructors are central to learning activities, directing the learning process and controlling students' access to information. Students are passive recipients of knowledge, rather than constructing their own knowledge. Thus, while objectivism promotes stability and certainty with respect to knowledge transmission, there are few opportunities for learners to express their own ideas, because objectivist teaching does not engage the mind appropriately to go beyond prior knowledge. The objectivist model is therefore criticized for stimulating surface learning and knowledge reproduction (Nunes & McPherson, 2003).

The constructivist paradigm of learning clearly diverges from the objectivist model which presumes that knowledge can be put directly into the learner's head. Constructivism regards learning less as the product of passive transmission than a process of active construction. Learning is considered as an active process in which learners construct new knowledge based upon their prior

knowledge. Constructivism is learner-centered, assuming that learners learn better if they construct knowledge for themselves, rather than being told by an instructor. Hence, given the shortfalls of objectivist, teacher-directed instruction, constructivism may be helpful to create a learner-centered environment that promotes active, independent and self-reflected knowledge construction (Duffy, Lowyck,, & Jonassen, 1993; Gros, 2002; Honebein, Duffy, & Fishman, 1993; Stefe & Gale, 1995; Wilson, 1998).

Constructivism is grounded in several philosophical traditions with three orientations: individual constructivism (Piaget, 1969, 1971), social constructivism (Vygotsky, 1978) and radical constructivism (Von Glasersfeld, 1994). These approaches are not mutually exclusive. They are closely related to each other. Whereas individual constructivism concentrates on individuals and their learning, social constructivism focuses on groups and their learning within socio-cultural contexts. For radical constructivism there is no knowledge independent of that constructed by learners, because knowledge is based upon constructions that are not tied to any external reality.

The exploration of the concept of constructivism in information technology education resulted in a highly diverse body of practical examples (Ben-Ari, 1998; Ben-David Kolikant, 2001; Booth, 2001; Fowler, Armarego, & Allen, 2001; Hadjerrouit, 1999; Parker & Becker, 2003; Pullen, 2001; Soendergaard & Gruba, 2001; Van Corp & Grissom, 2001).

Given this diversity, it is essential to focus on a set of fundamental principles, across all three types of constructivism (Matthews, 2002; Piaget, 1969, 1971; Spivey, 1997; Steffe & Gale, 1995; Von Glasersfeld, 1994; Vygotsky, 1978)

1. *Knowledge is constructed by using learners' prior knowledge as foundation.* Learning begins with the student's prior knowledge, because understanding a student's behavior requires an understanding of the student's prior knowledge.
2. *Knowledge is constructed by learners, not passively transmitted by teachers*. Knowledge construction stresses the crucial relationship between new knowledge and prior knowledge, since learners can only understand what they construct themselves. Learning develops through assimilating new knowledge into the learners' cognitive structures.
3. *Learning should foster student autonomy***.** Learning should foster self-awareness of all aspects of one's learning. This means that learning should place responsibility in students' hand to foster ownership, individual and group autonomy, initiative, and choice in learning activities.
4. *Learning requires cognitive skills.* Knowledge should be structured around concepts, problems, questions, and interrelationships in order to be useful for problem solving. The process of constructing interrelated knowledge structures requires cognitive skills, such analysis and reasoning skills, and meta-cognitive skills, such as reflection, self-evaluation, and group reflection.
5. *Learning should be embedded in authentic tasks*. Rather than applying knowledge to solve abstract problems, learning should focus around a set of realistic, intrinsically motivating problems that are situated in some meaningful real-world tasks. Real-world problems have enormous potential for learning because knowledge construction is enhanced when the experience is authentic.
6. *Learning should foster dialogue, negotiation and collaboration.* As a learner gains experience in a social situation, this experience may verify a learner's knowledge structures or it may contradict those structures. Therefore, learning should foster interactive questioning, negotiation, and group dialogue. It is thus important to include a broad community of audiences beyond the instructor.

7. *Learning should be assessed formatively*. Formative assessment is embedded in the learning process, focuses on authentic tasks and problems with challenges. In addition, formative assessment should focus on students' individual cognitive development and metacognitive skills, as well as collaboration and group work.
8. *Teachers serve as facilitators of learning, not transmitters of knowledge.* In a constructivist setting, teachers serve primarily as guides and facilitators of learning, not as transmitters of knowledge. It follows that teachers must learn how to understand students so that they can interpret responses better and guide communication more effectively in order to facilitate learning.
9. *Learning activities.* Traditional modes of teaching, such as lectures, should be replaced by a set of learning activities where the students are actively engaged in the knowledge being constructed.

Designing a Pedagogical Model

Constructivism is a theory of learning, not a description of instructional techniques. But the general principles of constructivism may be helpful to design a pedagogical model to support active. A pedagogical model is a conceptual construct that can be used by teachers as a framework for instruction using a specific learning theory (Nunes & McPherson, 2003). Thus, the starting point for developing a pedagogical model is the constructivist learning theory and research work related to the design of constructivist learning environments (Bradley & Oliver, 2002; Duffy, Lowyck, & Jonassen, 1993; Gros, 2002; Hirimi, 2002; Honebein, 1998; Honebein, Duffy, & Fishman 1993; Wilson, 1993).

The pedagogical model attempts to operationalize constructivist principles. It encompasses a learning management process and many learning phases. The goal of the learning management process is to enable instructors to monitor the learning process through providing feedback, replying to e-mail, communicating, asking students, etc. The learning phases are an orderly set of interdependent learning activities moving from context analysis to the communication of learning results. The model consists of eight phases (Figure 1).

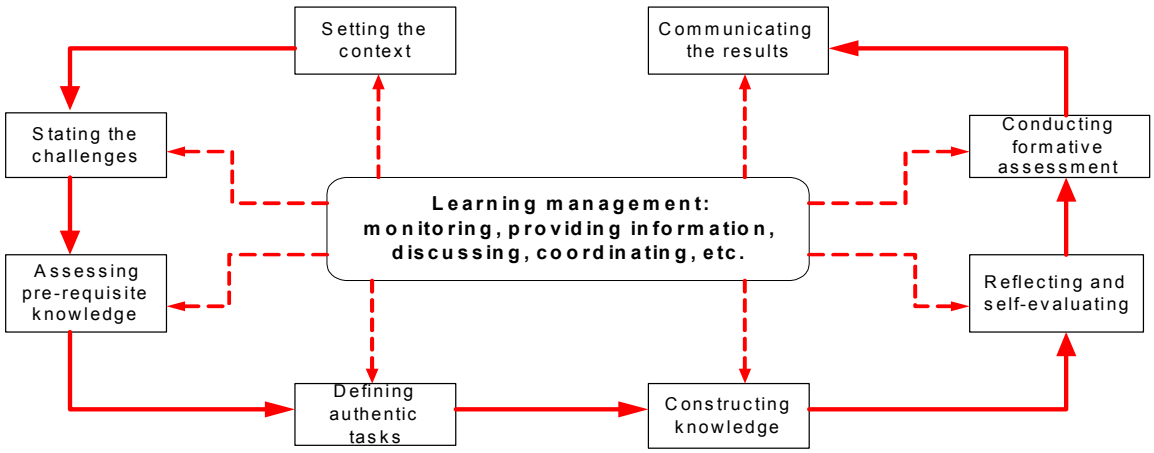

Figure 1: Pedagogical Model

Managing the Learning Process

This activity occurs throughout the whole learning process. To manage the learning process, instruction is reformulated as a dialogue between teachers and students. Dialogue involves social

interaction, negotiation, consensus building, and eventually conflict resolution. The role of the teacher in this process is not to dictate solutions, but rather to participate in discussions by monitoring students' work, facilitating and supporting the students in their search and supply of relevant information, coordinating the students' milestones of their projects, providing information and responses to queries, managing students' time effectively, negotiating learning goals and project tasks with the students according to their interests and motivations, etc.

This process refers also to the fact that a group of students work together and collaborate on a task. Collaboration is essential students because they are more likely to become engaged in the learning process when they must work together. Such collaboration should take place within the context of a team. Each member of a team must be assigned a role with team responsibilities. A collaborative team can be as small as two students, and in most cases not be larger than three or four students. The organization of Web engineering projects in small groups allows the supervision to be achieved through dialogue, discussions and regular group meetings in collaboration with the instructor.

Setting the Learning Context

A crucial concern of constructivist learning is the definition of the context in which learning takes place. Hence, this phase is a process of gathering data from the real environment. Basically, the context can be determined through the identification of the elements that directly influence the learning process: the subject matter and the associated skills, the instructor(s), the learners, the pedagogy, the technology, and the organizational and institutional climate.

First, the context is affected by the current state of the subject matter, the topics and the associated skills. Second, the context obtains information both from the instructors and the students. Additionally, the context is affected by learning theories and pedagogical strategies. These form the very basis of any learning environment. Then, the context is affected by the information technology being used, such as the technical platform and online resources, etc. Finally, the context is affected by organizational and institutional aspects, such as legal, ethical, and security considerations.

Stating the Challenges and Learning Goals

Web engineering is facing a broad range of challenges: constantly evolving Web technologies, evolutionary development, versatility of users' requirements, non-engineering issues, etc. By setting the challenges, the instructor can help ensure that students acquire Web engineering skills, while allowing them to take different paths toward achieving the learning goals based on their prior knowledge and experience. The primary challenges and learning goals are (Hadjerrouit, 2003b):

1. To enhance the comprehension of Web engineering development processes.
2. To use the Unified Modeling language (UML) for modeling Web-based applications.
3. To carry out Web engineering projects using the object-oriented development methodology.
4. To use a variety of Web technologies to construct Web-based applications.
5. To modify, extend, and eventually reuse knowledge from various sources.
6. To acquire communication, reading, and writing skills.
7. To reflect, evaluate, explain, and justify Web engineering solutions.
8. To understand ethical, legal, cultural, and social dimensions of Web engineering.

Assessing the Students' Prerequisite Knowledge and Skill Level

This phase serves to assess the students' prerequisite knowledge and skill level to meet the challenges of the Web engineering course. Before entering the field of Web engineering, it is expected that most students have taken courses in the following subjects during the first five semesters of the Bachelor Study Program in Computer Science at the Faculty of Mathematics and Sciences. Students should be able to use Web technologies and authoring tools that permit the implementation of Web-based applications, such as HTML, JavaScript, CGI Script, and PHP (Hadjerrouit, 2004); object-oriented modeling and programming with UML, Java, and Micosoft Visio Professional. Second, they should be able to apply client-server architectures to build Web-based applications, in particular Web databases with the JDBC, Java Servlets and MySQL (Hadjerrouit, 2003a).

Negotiating Authentic Tasks

After setting the learning goals, and assessing the students' pre-requisite knowledge and skill level, the next step consists of negotiating authentic tasks with the students and how to perform them according to their abilities and pre-requisite knowledge. The definition of authentic tasks is an iterative process with revisions.

Basically, authentic tasks are those based on the world of experience and work. En authentic task is a realistic, intrinsically motivating problem that is situated in some meaningful real-world environments in order to get students actively involved in the learning process. The students are the initiators of defining the tasks. There are many factors that influence the definition of an authentic task. First, the task must be related to Web engineering applications. Second, the task must be linked to an external organization as the students can explore the real-world dimension of the task. Hence, the task must contain all the relevant information that surrounds the problem to be solved, the stakeholders, organizational aspects, etc. Third, the task must be motivating so that the student is encouraged to solve it.

Authentic tasks may be Web applications, such as product catalogues, electronic shopping, Web stores, electronic books and digital libraries, academic Web sites, travel entertainment sites, electronic banking, etc. (Ge & Sun, 2000; Hadjerrouit, 2001). These applications are realistic and motivating for most students, largely because of their popularity and marketability.

The teacher takes on the role of facilitator of this activity. She/he is responsible for helping the students to specify their tasks. The problem is that students with little prior knowledge may have difficulties determining their tasks. Hence, it may be necessary to provide access to a set of similar projects and related cases that the students can draw on to represent their deficient experience.

Constructing Web-Based Applications

This phase is concerned with the construction of Web-based applications. Students work in groups to acquire Web engineering skills throughout the development process. The phase is time consuming and demanding in terms of efforts. The process of knowledge construction is discussed in section "Web Engineering: Knowledge Construction".

Reflecting and Self-evaluating

During the development process, students benefit from reflecting on how they have achieved the learning goals, before they submit their reports to the instructor. What difficulties were encountered? What cause the difficulties? How will they overcome them? What general principles may be extracted from the learning experiences? What patterns have they perceived in the problems

presented? Learning experiences often give students the necessary pieces of the "how-to-do-Web engineering", but many students will not attempt to fit the pieces together unless the instructor asks them specifically to do it. This phase thus helps students to develop their own ability to self-assess their own project work.

Conducting Formative Assessment

The way instructors assess students has far reaching consequences. Summative assessment is the attempt to summarize student learning at some point in time, for example the end of a course. Most standardized tests are summative, e.g. written exams. Summative assessment has advantages, but it should not be the only form of assessment (Lambert & Lines, 2000).

By contrast, formative assessment occurs when teachers feed information back to students in ways that enable the student to learn better, or when students can engage in a similar, self- reflective process. Formative assessment is asking questions in order to determine the learners' current understanding, so that they can make adjustments if necessary (Beverly & Bronwen, 2002). It is based on the principle that the evaluation of learning should not be separated from the learning process. Hence, assessment should be embedded in the learning process and spread out over the duration of the course.

Project-based work in Web engineering is particularly suited to formative assessment. To assess their project work, students submit three reports covering the analysis, design and implementation phases, and a final report covering the entire project. The instructor assesses the submitted reports and provides feedback, so that the students can improve them. When necessary, the reports are rewritten and submitted to the instructor once again. Hence, assessment occurs at least four times throughout the Web development process (Hadjerrouit, 2003b).

Presenting and Communicating the Results

In the last phase of the learning process, students are expected to formally communicate the results of their work orally to the whole class. During the entire development process students are communicating the results of their efforts in an informal way and discussing what they have learned with other students and the instructor. To formally communicate their results, students present their work, showing others what they have done and discussing what they learned. The instructor then grades the presentations based on the amount of work delivered during the course and the quality of the presentations.

Web Engineering: Knowledge Construction

This section is concerned with the specification of the knowledge construction phase from section "Constructing Web-Based Applications". During this phase, students spend considerable time and effort constructing Web-based applications throughout the development process. Learning Web engineering means constructing complex knowledge structures by connecting various types of knowledge to each other until the software solution has a coherent structure. Hence, the core of Web engineering development process is an integrated set of activities that are organized to maximize students' attention and motivation. Each activity has a name and a duration (the time required to perform the activity). To perform the activities according to constructivism (principle 4), students must learn to connect different, but closely interrelated, types of knowledge:

1. The prerequisite knowledge needed to do the activity in terms of concepts, skills, technology, and other relevant information. Students should be familiar with this knowledge before attempting the activity.

2. The knowledge that will be addressed during the activity and the expected outcomes. This component tells the students what they are expected to learn in terms of concepts, skills, and which steps they have to follow in order to solve the problem.
3. A problem situation in terms of a realistic, intrinsically interesting and motivating task that will be addressed during the activity.
4. Knowledge of solutions of previous problems and related cases that may be reused to solve the problem. Related cases enable students to examine previous solutions and relate them to the current problem.
5. Information banks that offer various types of information that might be used to obtain a greater understanding of skills and concepts, including online textbooks and documents.
6. Evaluation and reflection. After addressing the problem, students re-examine and reflect on their solutions and the criteria used to do this. This step is important because the reflection will reveal how and what the students have learned.

The development process contains five specific activities: analysis, system design, Web design, prototyping, and usability testing; and four generic activities occurring throughout the development process. Each activity is directly related to a skill described in Table 1. For example, analysis modeling activity is related to the analysis skill. Design activity is related to the design skill, etc. This paper concentrates on three specific and two generic activities.

Analysis Modeling Activity

Before attempting the analysis activity, students must already possess prerequisite knowledge in object-oriented modeling with UML and Microsoft Visio Professional. Furthermore, students must have understood the problem requirements. To perform this activity, students may reuse examples from textbooks and components from previous project solutions and related cases. Reuse requires the ability to recognize similarities and differences between the problem to be solved and previous problems and their solutions. Finally, they must be able to reflect on what they did using evaluation criteria. The expected outcome of this activity is an analysis report with a specification of the problem requirements using classes, use-cases, state, activity, and interaction diagrams. The following table gives an overview of the analysis modeling activity.

Table 2: Analysis modeling activity

Name of the activity: Analysis modeling.
Duration: Four weeks + one week for revision
Prerequisite knowledge, concepts, skills, technologies, and other relevant documents: 1) Microsoft Visio Professional. 2) Object-oriented modeling with UML (Unified Modeling Language): a) Class modeling. b) Use-case modeling. c) State change, activity, and interaction modeling. 3) Problem requirements.

Expected outcome: Analysis modeling report. The purpose of the analysis modeling is concerned with the specification of the problem requirements. There are five major steps: 1) Read the problem requirements. 2) Elaborate class specifications: - Discover classes. - Specify classes. - Model association relationships. - Model aggregation and composition relationships. - Model generalization relationships. 3) Elaborate behavior specifications using use cases: - Discover actors and use cases. - Specify actors and use cases. 4) Elaborate state change, activities, and interaction specifications. 5) Write the analysis modeling report.
Reusable solutions: - Students' project analysis solutions and reports from previous years. - Instructor's analysis examples and related cases. - Analysis examples and related cases extracted from textbooks.
Information banks: - Online course material. - Problem requirements. - Textbooks.
Problem situation: Online shopping of computers (This is just an example). The problem requirements are as follows: The customer can select a standard configuration or can build a desired configuration online. To place an order, the customer must fill out the shipment and payment information. The customer can check online at any time the order status. The ordered configuration is shipped to the customer together with the invoice. (…)
Evaluation criteria: Appropriateness, readability, completeness, and reliability of requirements.

Design Modeling Activity

The objective of the design modeling activity is twofold. First, the design of the system by addressing both architectural issues (client-server, packages, components, nodes) and detailed design issues (collaborations). Second, the elaboration of the analysis models developed in the previous phase, that is, adding detailed design and extending existing use case models to effectively turn them into design documents (Table 3). Before attempting this design activity, students must have performed the analysis activity. To design their systems, students may reuse examples from information banks, and eventually some components from previous project solutions and related cases. Reuse requires the ability to recognize similarities and differences between the problem to be solved and previous design solutions. Finally, students must be able to reflect on what they did using evaluation criteria. The expected outcome is a design report.

Table 3: Design modeling activity
Name of the activity: Design modeling.
Duration: four weeks + one week for revision

Prerequisite knowledge, concepts, skills, technologies, and other relevant information: (1) Microsoft Visio Professional. (2) Object-oriented modeling with UML: (a) Class packages, component, and deployment diagrams. (b) Collaboration diagrams. (c) Analysis modeling report. (3) Client-server architectures.
Expected outcome: Design report. The purpose of design modeling is concerned with the modeling of the system architecture and its components. There are eight major steps: 1) Read the analysis modeling report. 2) Design the overall system architecture using a client-server architecture. 3) Design the system architecture using class and use case packages 4) Design the logical components using component diagrams. 5) Design the physical components and nodes using deployment diagrams. 6) Design collaborations using collaboration and sequence diagrams. 7) Add design details to the existing uses case models from analysis. 8) Write the design report.
Reusable solutions: - Students' project design solutions and reports from previous years. - Instructor's design examples. - Design examples and related cases extracted from textbooks.
Information banks: - Online course information. - Analysis modeling report. - Textbooks.
Problem situation: Online shopping of computers. Based on the analysis modeling report from the previous activity, the objective of design modeling phase is twofold. First, the design of the system by addressing both architectural issues (client-server, packages, components, nodes) and detailed design issues (collaborations). Second, the elaboration of the analysis models developed in the previous phase, that is, add detailed design and extend existing models to effectively turn them into design documents.
Evaluation criteria: Abstraction, modularity, refinement, control hierarchy, information hiding, structure and logic.

Prototyping and Testing Activities

Prior knowledge from the programming language Java is required to perform this activity (Table 4). Web programming and Database development with the JDBC, Java Servelts, and MySQL, is also a prerequisite for anyone entering the field of Web engineering, since the implementation of a valuable software solution for a Web-based application depends on. Students must have performed analysis and design modeling before attempting this activity. It is recommended to reuse program code from previous projects, because it is not necessary to develop Web-based applications from the ground. The expected outcome is a piece of well-documented and tested software that implements the design model. The software solution must be based on the three-tier client/server architecture. Finally, students must be able to evaluate the solution using software

quality criteria. It is not expected to develop a sophisticated software solution, because of the very short time frame remaining for the implementation phase (3 weeks). The most time-consuming task is the graphical user interface, largely because its implementation requires more than the use of Web technology (Nielsen, 2000). Web implementation is a subtle combination of esthetical and cognitive issues. It requires an optimal balance between visual sensation, graphic information, text, and multimedia support.

Table 4: Prototyping, coding, and testing.

Name of the activity: Evolutionary prototyping and testing.
Duration: Two weeks + one week for revision.
Prerequisite knowledge, concepts, skills, technologies, and other relevant information: (1) Object-oriented programming with Java. (2) Web programming with HTML, JavaScript, PHP, CGI Script, or similar languages. (3) Database development with the JDBC, Java Servelts and MySQL. (4) Deployment tools and Web editors such as Macromedia Dreamweaver or similar. (5) Design modeling report.
Expected outcome: prototypes, program code and test strategies: The purpose of this phase is to implement the design solution from the previous phase using an evolutionary prototyping and incremental development methodology. The implementation consists of a seven steps: 1) Split up the program's execution logic between the client and the server processes. 2) Develop a prototype for the graphical user interface. 3) Develop a prototype for the database. 4) Develop a prototype for the interactions between the user interface and the database. 5) Refine the overall software solution. 6) Develop strategies for unit and integration testing. 7) Write an implementation report.
Reusable solutions: - Students' software prototypes and test strategies from previous projects. - Instructor's code examples and test guidelines. - Software solutions and examples extracted from textbooks.
Problem situation: Implementation of the design solution from the previous phase.
Information bank: - Online Java and MySQL program code. - Online course information and textbooks. - Design modeling report.
Evaluation criteria: • General criteria: Class cohesion and coupling, code readability, program structure and navigation, testability. • Web criteria: user control, navigation, consistency, feedback, aesthetics and usability.

Activities Occurring throughout the Development Process

Reuse of Components

One of the advantages of constructing Web-based applications is that it is not necessary to develop everything from the ground, since the reuse option is an essential element of Web engineering. Indeed, the reuse philosophy of Web engineering relies on the basic idea that Web-based projects are similar. Consequently, solutions of past project solutions may be modified and reused to meet the requirements of new projects. The reuse option is potentially very relevant for building e-commerce applications and instructional Web sites, since both have a number of similar components, e.g. user interface, shopping card, product catalogue for e-commerce applications; reusable learning objects for instructional Web sites. The reuse activity requires analogical reasoning, in particular recognizing similarities and differences between past and new projects.

Reading, Writing, and Communication

The consideration of communication, writing, and reading skills is a key issue of a the pedagogical model, because they enable students to think critically, to express a point of view, and to reflect one's own learning in the course of project work. The ability to communicate, write, and read both in Norwegian and English is clearly an advantage for any student because it is a prerequisite for a successful professional life. However, it may be difficult to learn these skills within a one-semester course, largely because their acquisition requires active involvement in Web engineering projects over a long period of time (Miller & Luse, 2004; Soendergaard, & Gruba, 2001).

The Evolution of the Pedagogical Model

The pedagogical model outlined in this paper evolved from an initial, simple approach to constructivism to a more sophisticated pedagogical model during the last 6 years through a process of experimentations, evaluations, and redesigns, where the shortcomings of each evolutionary stage were identified and improved. The evolutionary changes of the model are a general process that affects the course content, the learning goals, the role of the teacher and students, etc. (Pahl, 2003).

Since the academic year of 2000 the pedagogical model has evolved through the influence of five factors: the introduction of the object-oriented development methodology with the Unified Modeling Language (UML), online resources, students' evaluations, Web technologies, and the state of the constructivist learning theory.

First, the object-oriented development methodology with UML affects the pedagogical model. It enables the use of a common set of terms throughout the whole modeling process. Furthermore, because students are supposed to have sufficient prerequisite knowledge in object-oriented programming, it might be easier for them to move from object-oriented programming to object-oriented modeling with UML. In addition, UML facilitates the reuse of objects because of their similarities and collaborations. Finally, UML is favored in industry, and this is a major motivation for students for pragmatic considerations.

Second, the gradual evolution of the pedagogical model occurs through the use of online resources that students may modify and reuse to solve their project work. This occurs through the access to online resources that support active and self-reflective learning, in particular, students' project solutions from past versions of the course, reusable program code, well-designed analysis

and design modeling solutions that might be adapted, modified, and extended to meet the requirements of new problems. This use of online resources fosters the acquisition of reuse and analogical thinking skills, e.g. comparing, contrasting, and recognizing similarities and differences between new problem situations and previous solutions; as well as modifying and reusing existing solutions.

Third, the model evolves through successive students' evaluations instructor's reflections and observations. Students' evaluations by means of rigorous standard questionnaires are becoming an important tool for improving the quality of education, in particular since the implementation of the Quality Reform in Higher Education in Norway in 2003.

Fourth, the evolution of Web technologies affects the pedagogical model. For instance, advanced Web editors such as Macromedia Dreamweaver make the implementation of the user interface easier and reduce the amount of time required to achieve project work. This, in turn, helps students to focus more on analysis and design activities, and self-evaluation of their project reports.

Finally, the evolution of the pedagogical model is affected by the constructivist pedagogy itself through its use and experimentation. The pedagogical model is also improved through a greater understanding of constructivist principles by means of literature studies, journal articles, research reports and participation in international conferences and congresses.

To sum up, various factors influence the evolution of the pedagogical model by improving its overall design. The evolution of the pedagogical model occurred over a timescale of five years and will continue in the future. The current model seems to be more robust and better suited to the students' needs. This, in turn, affects the course material, the role of the instructor, the expectations of the students. For instance, the instructor gradually shifted from the teacher as the center of knowledge to the position as facilitator of learning. Dialogue with the students becomes then one the primary vehicles used to negotiate authentic tasks, and less the preparation of study material and information resources. Discussions are manageable with classes around 10-20 students and 3-5 student teams each semester. Hence, the change of emphasis from "teaching" to "facilitating and guiding" reduced the amount of lecture notes, overheads, handouts and lesson plans. But, in contrast, the instructor had to spend more time both online and face-to-face, collaborating with the students, asking and answering questions, providing feedback, assessing their learning, etc.

Experiences with Implementing the Pedagogical Model

Web engineering is taught during the sixth semester of the Bachelor Study Program in Computer Science at the Faculty of Mathematics and Sciences. The course is appropriate for undergraduate students. Since the implementation of the Quality Reform in Norway in 2003, the course should be taught in the spring semester, and no longer in the fall semester. As a result, the course was not taught in 2004 because of changes in the curriculum and course sequence.

As the approach was being attempted for the first time in 1999, it was hard to implement constructivist principles for the instructor, for two reasons:

First, a drastic change from objectivist, teacher-centered techniques to constructivist, learner-centered approaches is difficult for university faculty instructors who are not educational researchers. Constructivism is a challenge as the instructor must not just transmit information to students, but must act as a facilitator and guide of learning (Nunes & McPherson, 2003; Steffe & Gale, 1995). Constructivism is more demanding in terms of communication with students and provision of intrinsically motivating learning activities that are situated in real-world environments.

Second, Web engineering in 1999 was in the very beginning of its development. It was therefore not obvious which skills, technologies, techniques, and tools one would teach to educate skilled Web engineers. As a result, constructivist principles could not be applied to Web engineering as originally anticipated. Hence, the academic years of 2000 and 2001 were used to gain more experience in Web engineering. The instructor had to test various Web technologies, techniques and tools, introduce the object-oriented software development methodology, contact software organizations, discuss with Web administrators and publishers in order to define authentic tasks for the students. Also important was the instructor's participation in international conferences related to information technology and computer science education, the reading of journal articles related to Web engineering and constructivist learning. It was also important to choose the right textbooks and study material, develop learning activities based on well-articulated constructivist principles, etc.

The model described in section "Designing a Constructivist Learning Model" was applied for the first time in the fall semester of 2002. Before the first week of a 15-week semester, and depending on the circumstances (students' profiles, physical, institutional, and organizational environment, etc.), the instructor conducts a context analysis by defining the learning environment through the identification of the contextual elements that directly influence the learning process: The current state of the subject matter, learners' characteristics, the state of the pedagogy, the available resources, and the organizational and institutional climate. By setting the context, the teacher provides information about the learning environment for the current semester. During the first week, the instructor presents the approach to the students, sets the learning challenges for the course, and collects students' prerequisite knowledge by means of a questionnaire. The results of the questionnaire are used to determine the skill level of the students. Once placed within the context and given a certain direction, the focus shifts to the students. Then, the next step consists of negotiating project tasks with the students and how to perform them according to their needs and motivations. The definition of authentic tasks is an iterative process with many revisions, and it can take up to two weeks. Then, the students spend a considerable amount of time (12-13 weeks) performing Web engineering tasks, before submitting their work for formative evaluation, and, finally, presenting and communicating their results to the whole class.

Evaluation Goal, Methods, and Results

The goal of the evaluation is to study the effects of constructivism and associated pedagogical model on students' learning. In an attempt to provide a consistent evaluation of the learning process, the author advocates a combination of three evaluation methods:

First, to measure learning outcomes, the university requires the use of methods based on a six-point scale from 1 to 6, where 6 is coded as the lowest, 1 as the highest, and 4 is required to pass the subject matter. Obviously, this method does not automatically encourage students to think constructively when used in written exams, because it is quite possible to pass an exam without using constructivist problem-solving techniques. In constructivist learning environment where the emphasis is the acquisition of critical skills, summative assessments like written exams are clearly not consistent with learning that takes place in those environments (Salomon & Perkins, 1998). However, this method, when applied to project-based work, may be appropriate, because the evaluation of project work is embedded in the learning process and spread over the duration of the course. In addition, the emphasis of project-based work is on the acquisition of critical skills. This is in line with constructivism.

Second, assessment by means of a standard questionnaire was used to assess a greater portion of the learning environment, such as learning resources, intended learning objectives, prerequisite knowledge, project activities, skill acquisition, dialogue, etc. Obviously, standard assessment

methods have many advantages, but they are not entirely sufficient to evaluate constructivist learning and the associated pedagogical model. Therefore, they should not be the only form of assessment.

Finally, the evaluation of the pedagogical model included issues of qualitative character that are of particular significance from a constructivist point of view, such as motivational aspects, effort and time required to perform project activities, students' learning difficulties, teamwork and collaborative learning, etc. Many of these issues may be difficult to evaluate with standard assessment methods alone. Evaluation methods, such as semi-structured interviews with the students, teacher's observations in the classroom, students' comments to open ended questions, are more appropriate to evaluate qualitative issues.

Project Work Assessment

In line with constructivism, assessment of students' project work is embedded in the learning process and spread out over the duration of the course. The evaluation consisted of assessing

a) The quality of the submitted project reports,

b) The presentation of project results to the whole classroom, and

c) The active participation of the students in project work.

Table 5 shows project work assessments the last five years.

Table 5: Project work assessments 1999-2003 : Average grades

Academic year	Average score	Number of student teams
2003	1.79	3
2002	1.87	3
2001	2.14	5
2000	2.80	5
1999	2.70	3

Table 5 shows that students in 2002 and 2003 performed better than in 2001, 2000 and 1999. For instance, two teams in 2002 were able to write substantial analysis and design reports (over 450 pages), implement Web-based applications with advanced functionalities, and test them adequately. In addition, all teams managed to submit three well-structured reports covering the analysis, design, and implementation stages, and the final report documenting the final project work.

A possible interpretation of the learning outcomes is that project-based work seems to encourage students to think constructively in order to successfully perform project activities, because the emphasis is not on factual recall and memorization of facts or the mastery of a test. Thus, it may very well be that students felt that constructivist-oriented learning activities are appropriate and motivating to achieve deeper learning.

Students' Responses to Standard Questionnaires

To assess a greater portion of the learning environment, the evaluation was extended to include important aspects of constructivism and associated pedagogical. This evaluation was investigated in terms of students' responses to a standard questionnaire. To measure the responses, a five-point Likert scale from 1 to 5 was used, where 1 was coded as the highest and 5 as the lowest (1 = "Strongly Agree", 2 = "Agree", 3 = "Neutral", 4 = "Disagree", 5 = "Strongly Disagree"). The evaluation was organized into three categories of criteria that are related to the pedagogical model:

1. Prerequisite knowledge, learning objectives, course resources, and level of difficulty.
2. Motivation, project activities, profession of Web engineering, skill acquisition.
3. Feedback, interactions, dialogue and collaboration with the instructor.

Each of these categories addresses many issues. Category 1 addresses issues 1-4. Category 2 is concerned with issues 5-8.Category 3 deals with issues 9-11. The criteria and the associated issues were (Table 6):

Table 6: Evaluation criteria and associated issues

Evaluation criteria	Associated evaluation issues
Prerequisite knowledge	1. The course framework takes in consideration my own background knowledge.
Learning objectives	2. The intended learning objectives of the course framework are achieved.
Course resources	3. The course resources provide appropriate support to construct my own knowledge in Web engineering.
Level of difficulty	4. The knowledge level of the course (level of difficulty, scope, and depth) is appropriate.
Motivation	5. The course framework supports my motivation and involvement in Web engineering.
Project activities	6. The course framework focuses on authentic project tasks that might be used in the real world.
Profession of Web engineer	7. The course framework is important for my future carrier as Web engineer.
Skill acquisition	8. Project activities enable me to acquire Web engineering skills at various stages throughout the development process.
Feedback	9. Feedbacks from the instructor help me identify the strengths and limits of my knowledge.
Interactions	10. There are a variety of opportunities (e-mail, face to face, classroom, lab, etc.) to interact with the instructor.
Dialogue & collaboration	11. The instructor encourages discussion, collaboration and dialogue.

The students were asked to respond to the questionnaire by placing a cross "X" in the appropriate box using the scale provided. Tables 7, 8, and 9 provide total responses as numbers. Nine students out of the 12 registered for the course responded to the standard questionnaire. The results for the year 2002 are shown in Table 7, 8 and 9. Data collected in 2003 by means of the same questionnaire corroborate those in 2002.

Table 7: Evaluation criteria: Prerequisite knowledge, learning objectives, course resources, and level of difficulty.

Evaluation criteria	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1. Prerequisite knowledge	2	3	3	1	0
2. Learning objectives	1	5	2	1	0
3. Course resources	2	5	2	0	0
4. Level of difficulty	0	4	4	1	0

Table 8: Evaluation criteria: Motivation, project activities, profession of Web engineering, and skill acquisition.

Evaluation criteria	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
5. Motivation	2	5	2	0	0
6. Project activities	2	6	1	0	0
7. Profession of Web engineering	1	5	2	1	0
8. Skill acquisition	1	5	2	1	0

Table 9: Evaluation criteria: Feedback, interactions, dialogue and collaboration with the instructor.

Evaluation criteria	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
9. Feedback	2	5	2	0	0
10. Interactions	1	6	2	0	0
11. Dialogue & collaboration	2	5	1	1	0

Analysis of the responses to the categories "Strongly Agree" and "Agree" show that students were globally positive about the issues that were evaluated by means of a standard questionnaire. Considering, in addition, that some of the neutral responses can be regarded as a tacit approval of the category "Agree", otherwise the students would have chosen the category "Disagree", one can be satisfied with the evaluation results. Thus, it appears that some important aspects of the peda-

gogical model were quite well implemented. However, one should be careful to draw far-reaching conclusions from this kind of summative assessment that was conducted at the end of the course. Hence, the responses to the standard questionnaire alone may not be sufficient to give a correct picture of the strengths and weaknesses of the pedagogical model and process of learning.

Instructor's Observations and Students' Comments

Considering that project work assessments and responses to a standard questionnaire alone are not sufficient to give a reliable picture of the pedagogical model, the evaluation was extended to include issues of qualitative character that focus on the learning process.

Data were collected by means of students' comments and responses to semi-structured interviews, and teacher's observations. This evaluation is based on experiences from 2002 and 2003.

The method used for data analysis consisted of finding diverse pieces of evidence from three different perspectives: teacher's perspective, students' perspectives, and, to some degree, the perspective of the research literature. To facilitate the analysis of the data collected, the evaluation criteria and issues of the semi-structured interviews were, to some degree, closely aligned both with the issues of the questionnaire (Table 6) and the teacher's observation criteria (Agostinho, & Herington, 2003).

Data analysis involved many techniques:

a) Defining different observations' criteria and associated evaluation issues that fit the pedagogical model.
b) Asking and interviewing the students what they think about the issues and what the instructor is observing.
c) Comparing the data that was collected earlier in 2000 and 2001 with data collected in 2002 and 2003.
d) When possible, finding evidence in the research literature that supports or contradicts the data collected.

Given these considerations, data collection and analysis focused on the following issues:

1. Skill acquisition process, efforts and time.
2. Collaboration, teamwork and communication skills.
3. Motivation, real-world project activities, and profession of Web engineering

Skill Acquisition Process, Efforts and Time

Students reported that analysis modeling was the most difficult skill. This finding was not surprising for the instructor, since most students had never performed analysis modeling before entering the field of Web engineering. In contrast, design modeling was relatively easier for most students. Some students believed that UML – through a common set of concepts throughout the whole modeling process – played a central role in moving from analysis to design modeling, because it does not require a radical change of the object-oriented techniques (Maciazeck, 2001; Stevens & Pooley, 2000). UML enabled them to design Web applications through a process of successive elaboration of an initial model.

However, because they encountered difficulties in the analysis phase, students suffered from the short timeframe remaining for prototyping, coding, and testing (3 weeks), resulting many times in late night work. But this did not produce major problems for the delivery of the implementation report, because the instructor did not require the delivery of a software system with advanced

functionalities, but just a well-documented prototype that realizes the basic functionality of Web applications. Many students also mentioned that the reuse of project components from previous versions of the course available online helped them to improve the quality of the reports.

Furthermore, some students reported that the elaboration of the analysis document, and, to some degree, the design report, took more time than expected. Because the documentation of analysis modeling was not rigorously planned and prepared, they did not manage to write a stable document but were forced to change it many times. As a result, project work had to be adjusted several times due to some misjudgments about time consuming, particularly in the analysis phase, which was often longer than planned. This led to some problems during the design and coding phase and some deficiencies in the resulting documentation. Students reported that they spent more hours per week than the time allocated to the project activities. Some estimated that they spent significantly more time with their projects than all other courses.

Teamwork, Collaboration, and Communication Skills

Teamwork, collaboration, and communication are complex processes. They involve working with students with different backgrounds and experiences, sharing out project tasks, organizing project activities, collecting and distributing information, writing up protocols of meetings, nominating a leader, expressing points of views, and many other issues which affect the group decision process (Blake, 2003; Miller & Luse, 2004). It is thus evident that in order to achieve effective collaboration in a team, students must be trained in teamwork and communication skills before and during the performance of project activities, because "a random collection of students does not necessarily make for an effective team" (Frank, Lavy, & Elata, 2003, pp. 285).

The complexities of teamwork made it difficult to teach these skills within a one-semester course, because of the high workload of the projects and the short time frame. Thus, given the diversity of students' prerequisite in teamwork, it was not expected that all students would be able to improve their collaborative skills very much. Likewise, the high workload of the projects and the short time frame were major problems that made it difficult to significantly improve communication skills within a one-semester course (Soendergaard & Gruba, 2001).

Given these considerations, it was not surprising to observe that some teams were not prepared to achieve successful collaboration, because they had difficulties handling problems between members of their teams, resulting sometimes in frustration and insufficient collaboration, and this affected the effectiveness of their teams. The instructor had to intervene in order to help those teams to overcome their problems. Fortunately, the teams performed better when the teacher provided help and support. However, when not prompted by the teacher, those teams tended not to solve their problems by their own. As a result, the instructor had to spend considerable time both online and face-to-face in order to provide proper help to those students, especially in the analysis phase and some weeks before the delivery of the final project reports.

Furthermore, students with difficulties defining authentic project tasks felt that it is the instructor's responsibility to define project tasks and learning objectives for them. Fortunately, after the first month, those students no longer expressed the same discontent and disappointment with the course even though they had to spend considerable time with their projects. At the end, they felt more confident in their ability to meet the course requirements, and were, to some degree, satisfied that the amount of time and effort put into coursework was worthwhile. But, it was sometimes not easy for the teacher to maintain lines of communication with some students, because of frustration related to the considerable amount of work, and problems with articulating and communicating their thoughts. Fortunately, when adjusting the help to each student and group of students, the instructor felt that those who had initially been weaker in the analysis phase had improved during the course, and had achieved relatively good scores.

Motivation, Real-World Project Activities, and Profession of Web Engineering

A major goal of the pedagogical model is to try to be as close as possible to reality. But still, in contrast to real-world projects from the industry, students' software products and reports must be developed and delivered within three months – just enough to go through a full life cycle of small software projects. Therefore one could not expect students to design a software product with advanced functionalities. Thus, the learning goal was to let students experience the challenges of developing small, but well-structured and documented software.

Despite the limited timeframe and the constraints of the course, most students were very satisfied with the real-world character of the projects. They felt that the course increases their motivation to learn and to make greater efforts in the right direction. Many students reported that their motivation is directly related to the marketability of the projects, the profession of software engineering, and future professional career. When asked about the relevance of marketability for their professional career, students were very clear in their responses. They believed that Web engineering combined with real-world projects is highly relevant to them as it can help them to gain practical experience and to increase their marketability. Thus, the motivational aspect should not be underestimated, since marketability seems to be very important for students.

As a result, it appears motivation is one of the most important factors for project's success. High motivation of all participants in a team resulted, in most cases, in god working atmosphere, which was very positive for the performance of project activities. Two students reported that this course was the most exiting, motivating, and useful one in which they had participated. This experience is consistent with the constructivist point of view which asserts that real-world projects are motivating to get students actively involved in knowledge construction and skill acquisition. Clearly, motivating students is an essential element of constructivist learning (Green, 1998, Hadjerrouit 1999, 2001).

Lessons Learned: Potentialities and Limitations of the Model

According to the evaluation results, it is obvious that teaching Web engineering proved to be more complex than simply applying the constructivist learning theory. So far, the lessons learned are as follows:

1. The pedagogical model gives teachers a framework for understanding students' learning process. It can help teachers to expose students to many aspects of the subject matter that are of crucial importance for students. It focuses on what really matters for the students: the acquisition of critical skills, authentic tasks, motivation, teamwork, collaboration and negotiation, formative assessment, and self-evaluation.
2. The evaluation results indicate that the model is effective in terms of motivating students and providing real-world project tasks. Thus, it is important to try to be as close as possible to reality. A motivating environment makes students actively involved in knowledge construction. Clearly, motivation is one of the most important factors for project's success. High motivation of all participants in a team results, in most cases, in god working atmosphere, which is very positive for the performance of project activities.
3. Even if it is impossible to draw any general conclusions from the students' experiences, it is clear that a significant number of students were positive about the implications of the pedagogical model. The model appears to encourage students to think constructively and become involved in Web engineering through project-based activities. Clearly, without

denying the fact that considerable efforts and time were necessary to achieve the learning goals, the model has great potentiality for improving students' learning. It was proven to be beneficiary to the students within the specific context of the course.

4. The model is more appropriate for students with solid prerequisite knowledge. Students who exhibited good scores in project work assessment generally possess solid background knowledge in authoring tools and Web editors; Web programming with HTML, JavaScript, and PHP; object-oriented modeling and programming with Java and UML; database development with the JDBC, Java Servlets and MySQL. Moreover, the better prepared the students are for teamwork, the greater the possibility of collaborating and communicating.
5. The form of this model can be suggested for rather small groups around 20 students and 3-5 student teams, because one instructor would have difficulty handling larger groups. Thus, the model needs to be studied varied instructional settings, for example for larger groups over 20 students and more than one teacher to confirm and support the findings of this work.
6. According to the teachers' experience, the practice of constructivism may be a challenge for inexperienced instructors, because instructors, in a constructivist setting, do not just convey information or supply facts, but must act as mentors, facilitators, guides, coaches, and mediators of learning. Likewise, dialogue may be a challenging task for inexperienced teachers, because they must learn how to understand students so that they can interpret responses better, guide communication more effectively, and adjust the help to each student. Such a drastic change of attitude is difficult for any teacher, and certainly for teachers in information technology and computer science who are not educational researchers.
7. The model is not entirely appropriate for students with little prerequisite knowledge. It may then be necessary to use objectivist techniques for helping those students when they need transferable knowledge from the instructor. Objectivist techniques are suitable for students with little prior knowledge and difficulties defining authentic tasks. Depending on the circumstances, it seems that a blend of objectivist and constructivist techniques may be the most appropriate instructional strategy to give those students a certain direction, especially in the first weeks of the course, before the focus shifts to the students. The constructivist approach is flexible enough to integrate objectivist techniques in the learning process.

Concluding Remarks and Future Research Work

In summary, applying a constructivist approach to learning is an iterative and evolutionary process that progresses through a series of experimentations, evaluations, and redesigns over many years. Hence, significant changes in education require considerable time and effort. This work is thus a long-term design-based research work (Barab & Squire, 2004; Design-based Research Collective, 2003) on the application of the constructivist learning theory in information technology education. The pedagogical model will be further developed through continuous cycles of design, experimentations, evaluations, and research directions, in particular:

1. Explore further the potentialities of constructivism in order to refine the understanding of learning issues;
2. Improve the evaluation methods, data collection and analysis, and develop hypotheses, theories or "proto-theories" that help communicate relevant educational and didactical implications to information technology educators and practitioners.

3. Refine the learning activities of the Web engineering construction process and develop pedagogical patterns for analysis and design activities that may be reused to solve new, but similar Web engineering problems.
4. Improve teamwork, communication, and collaboration in order for the pedagogical model to be more effective.

References

Agostinho, S. & Herington, J. (2004) An Effectiveness Evaluation of Online Learning Environment. *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21-26, 3476-3481.

Balasubramaniam, R., Pries-Heje, J., & Baskerville, R. (2003). Internet software engineering: A different class of processes. *Annals of Software Engineering, 14*, 169-195.

Ben-Ari, M. (1998). Constructivism in computer science. *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, Atlanta, Georgia, February 25 – March 1, 257-261.

Ben-David Kolikant, Y. (2001). Gardeners and cinema tickets: High school students' preconceptions of concurrency. *Computer Science Education*, *11*(3), 221-245.

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of Learning Sciences, 13 (1)*, 1-14.

Blake, M.B. (2003). A student-Enacted Simulation Approach to Software Engineering Education. *IEEE Transactions on Education*, *46*(1), 124-132.

Bleek, W-G., Jeenicke, M., & Klischewski, R. (2004). e-Prototyping: Iterative analysis of web user requirements. *Journal of Web Engineering, 3*(2)**,** 77-94.

Booth, S. (2001). Learning computer science and engineering in context. *Computer Science Education, 11*(3), 169-188.

Beverly, B. F & Bronwen, C. (2002). *Formative assessment and science education.* London: Kluwer Academic.

Bradley, C. & Oliver, M. (2002). The evolution of pedagogic models for work-based learning within a virtual university. *Computers & Education*, *38*(1-3), 37-52.

Conallen, J. (1999). *Building web applications with UML*. New York: Addison-Wesley.

Design-Based Research Collective (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher, 32*(1), 5-8.

Duffy, T. M., & Lowyck, J. & Jonassen, D. H. (1993). *Designing environments for constructive learning.* Berlin: Springer-Verlag.

Escalona, M. J., & Koch, N. (2004). Requirements engineering for web applications -- A comparative study. *Journal of Web Engineering, 2*(3), 193-212.

Fowler, L., Armarego, J., & Allen, M. (2001). CASE- Tools: Constructivism and its application to learning and usability of software engineering tools. *Computer Science Education, 11*(3), 261-272.

Frank, M., Lavy, I. & Elata, D. (2003). Implementing the project-based learning approach in an academic engineering course. *International Journal of Technology and Design Education, 13*, 273-288.

Ge, Y., & Sun, J. (2000). E-commerce and computer science education. *SIGSCE Bulletin, 32*(1), 250-255.

Green, A.M. (1998). Project-based learning: Moving students toward meaningful learning. In L. P. Steffe & J. Gale, (Eds.). *Constructivism in Education*. New Jersey Lawrence Erlbaum Associates.

Gros, B. (2002). Knowledge construction and technology. *Journal of Educational Multimedia and Hypermedia, 11*(4), 323-343.

Hadjerrouit, S. (1999). A constructivist approach to object-oriented design and programming. *Proceedings of the 4th Annual Conference on ITiCSE*, Cracow (Poland), June 27 – July 1,1999, 171-174.

Hadjerrouit, S. (2001). Web-based application development: A software engineering approach. *SIGCSE Bulletin*, *23*(2), 31-34.

Hadjerrouit, S. (2003a). Web databases. From: http://fag.hia.no/kurs/inf2490

Hadjerrouit, S. (2003b). Web engineering. From: http://fag.hia.no/kurs/inf2470

Hadjerrouit, S. (2004). Web programming. From: http://fag.hia.no/kurs/inf100/www_docs

Hirumi, A. (2002). Student-centered, technology-rich learning environments (SCenTRLE): Operationalizing constructivist approaches to teaching and learning. *Journal of Technology and Teacher Education, 10*(4), 497-537.

Honebein, P.C. (1998). Seven goals for the design of constructivist learning environments. In B. Wilson (Ed.), *Constructivist learning environments: Case-study in instructional design* (pp 3-8). Englewood Cliffs, NJ: Educational Technology Publications.

Honebein, P. C., Duffy, T. M. & Fishman, B. (1993). Constructivism and the design of learning environments: Context and authentic activities for learning. In T. M. Duffy, J. Lowyck, & D. H. Jonassen (Eds), *Designing environments for constructive learning* (pp. 88-108). NewYork: Springer-Verlag.

Lambert, D. & Lines, D. (2000). Understanding assessment: Purposes, perceptions, practice. London: Routledge-Falmer.

Lowe, D. & Eklund, J. (2002). Client needs and the design process in web projects. *Journal of Web Engineering, 1*(1), 23-26.

Matthews, M. R. (2002). *Constructivism and science education: A further appraisal*. Journal of Science Educational Technology, Vol. 11, no. 2, 121-134.

Maciaszek, L. A. (2001). *Requirements analysis and system design: Developing information systems with UML*. New York Addison-Wesley.

Miller, R. A., & Luse, D. W. (2004). Advancing the curricula: The identification of important communication skills needed by IS Staff during systems development. *Journal of Information Technology Education, 3,* 117-131.

Murugesan, S., Deshpande, Y., & Hansen, S. (1999a): Skill hierarchy for web information system development. *First Workshop on Web Engineering, at the International Conference on Software Engineering (ICSE),* 16 -17 May 1999, Los Angeles, USA.

Murugesan, S., Deshpande, Y., & Hansen, S. (1999b). Web engineering. Beyond CS, IS, and SE – An evolutionary and non-engineering perspective. *First Workshop on Web Engineering, at the International Conference on Software Engineering (ICSE),* 16 -17 May 1999, Los Angeles, USA.

Murugesan, S., & Ginige, A. (2001): Web engineering: An introduction. *IEEE Multimedia, 8*(1), 14-18.

Nielsen, J. (2000). *Designing web usability: The practice of simplicity.* New York New Riders.

Nunes, M. B., & McPherson, M. (2003). Constructivism vs. objectivism: Where is difference for designers of e-learning environments? *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03),* 496-500.

Pahl, C. (2003). Managing evolution and change in web-based teaching and learning environments. *Computer & Educations, 40*(2), 99-114.

Parker, J. R., & Becker, K. (2003). Measuring effectiveness of constructivist and behaviorist assignments in CS102. *SIGCSE Bulletin, 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2003)* (pp. 40-44), June 30 – July 2, 2003, Thessaloniki, Greece.

Piaget, J. (1969). *Judgment and reasoning in the child.* London: Routledge & Kegan Paul.

Piaget, J. (1971). *Genetic epistemology*. New York: W.W. Norton.

Pressman, R. S. (2001). "What a tangled web we weave". *IEEE Software*, *18*(1), 18-21.

Pullen, M. (2001). The network workbench and constructivism: Learning protocols by programming. *Computer Science Education*, *11*(3), 189-202.

Reichgelt, H., Lunt, B., Ashford, T., Phelps, A., Slazinski, E., & Willis, C. (2004). A comparison of baccalaureate programs in information technology with baccalaureate programs in computer science and information systems. *Journal of Information Technology Education, 3,* 19-34. Available at http://jite.org/documents/Vol3/v3p019-034-098.pdf

Salomon, G., & Perkins, D. (1998). Individual and social aspects of learning. In P. Pearson & I.-Nejad (Eds.), *Review of Research in Education*, *23*, 1-24. Washington, DC: American Educational Research Association.

Seffah, A., & Grogono, P. (2002). Learner-centered software engineering education: From resources to skills and pedagogical patterns. *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02),* 14-21.

Soendergaard, H., & Gruba, P. (2001). A constructivist approach to communication skills instruction in computer science. *Computer Science Education*, *11*(3), 203-209.

Spivey, N. N. (1997). *The constructivist metaphor: Reading, writing, and the making of meaning*. Academic Press.

Steffe L. P. & Gale J. (Eds.) (1995). *Constructivism in education.* Lawrence Erlbaum Associates.

Stevens, P., & Pooley, R. (2000). *Using UML: Software development with objects and components*. London: Addison-Wesley.

Van Corp, M. J. & Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education*, *11*(3), 247-260

Von Glasersfeld, E. (1994). *Radical constructivism in mathematics education*, Kluwer Academic Publishers.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* Cambridge. MA: Harvard University Press.

Whitehead, E. J. (2002). A proposed curriculum for a masters in web engineering. *Journal of Web Engineering*, *1*(1), 18-22.

Wilson, B. G. (Ed.) (1998). *Constructivist learning environments: Case studies in instructional design*. Englewood Cliffs, NJ: Educational Technologies Publications.

Biography

Said Hadjerrouit received the MS and PhD degrees in Software Engineering and Artificial Intelligence from the Technical University of Berlin (Germany), in 1985 and 1992, respectively. He joined Agder University College, Kristiansand (Norway) in 1991. He is currently an Associate Professor of Computer Science at the Faculty of Mathematics. He has been in the teaching profession for 22 years. He has extensive experience teaching object-oriented programming, Web design, database development, and software engineering. His research interests include computer science and software engineering education, didactics of informatics, E-Learning, Web engineering, and object-oriented software development with the Unified Modeling Language (UML). Hadjerrouit has published over 30 papers in international journals and conference proceedings.