

# A Blended Learning Model in Java Programming: A Design-Based Research Approach

*Said Hadjerrouit*  
*Agder University College, Kristiansand, Norway*

[Said.Hadjerrouit@hia.no](mailto:Said.Hadjerrouit@hia.no)

## Abstract

Blended learning, which blends face-to-face learning and online learning, is becoming an attractive model in higher education as new innovative information technologies are becoming increasingly available. However, just blending face-to-face learning with information technologies cannot provide effective teaching and efficient solutions for learning. To be successful, blended learning must rely on solid learning theory and pedagogical strategies. In addition, there is a need for a Design-Based Research approach to blending learning through successive cycles of experimentations, where the shortcomings of each cycle are identified, redesigned, and reevaluated. This paper reports on a study conducted on a blended learning model in Java programming at the introductory level. It presents the design, application, and evaluation of the approach and its implications for the learning of introductory computer programming.

**Keywords:** Blended learning, Design-Based Research, face-to-face learning, introductory programming, Java programming, learning cycle, online learning.

## Introduction

Educators generally agree that programming is a difficult matter, because it is more a skill than a body of knowledge. It is therefore hard for novice students to acquire programming skills within a one-semester course. Pedagogical approaches, which take advantage of learning theories and information technologies, have been proposed in the research literature to tackle the learning problems associated with introductory computer programming. However, there are very few evidence-based experiences, and the difficulty of learning introductory programming for novice students still remains.

As blended learning becomes more and more pervasive in higher education as the most prominent delivery mechanism (Bonk & Graham, 2006), expectations for learning benefits in computer programming are becoming greater. Hence, the underlying hypothesis of this work is that blended learning has the potential to improving the learning of computer programming. But, just providing educators with a mix of face-to-face learning and online learning, will not have the desired

---

Material published as part of this publication, either on-line or in print, is copyrighted by the Informing Science Institute. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission and payment of a fee. Contact [Publisher@InformingScience.org](mailto:Publisher@InformingScience.org) to request redistribution permission.

effect, if the underlying blended learning model does not rely on solid learning theory and pedagogical principles, on the one hand, and developed through successive cycles of analysis, design, implementation, and evaluation in varied contexts, on the other hand. Clearly, when implementing blended learning in computer programming, it is very important to be clear about the underlying pedagogical assumptions. A model of

blended learning needs to demonstrate which pedagogical principles and learning theories are operating. That is to say, progress in blended learning will come from a better understanding of the learning process and not automatically from improved technology (Nocols, 2003). Therefore, learning theories and associated pedagogical strategies must be one of the driving forces behind blended learning design.

In addition, to gain theoretical and practical insights, and generate some evidence-based claims about students' learning experiences in computer programming, the model must be explored in details and depth through the iterative and continuous cycle of design, evaluation, and redesign in varied educational contexts. Traditional approaches that separate design from evaluation, or where research is conducted after the design, are limited to achieve valuable results (Wang & Hannafin, 2003). A promising approach to blended learning in programming is the Design-Based Research paradigm.

The remainder of this article is organized as follows. First, the paper gives an overview of the Design-Based Research approach to blended learning. The second section specifies the research hypothesis and associated issues of the work. In the next sections, the paper outlines a blended learning model and applies it to an introductory programming course in Java. Finally, the summary of evaluation findings and implications for the design and delivery of blended learning conclude the article.

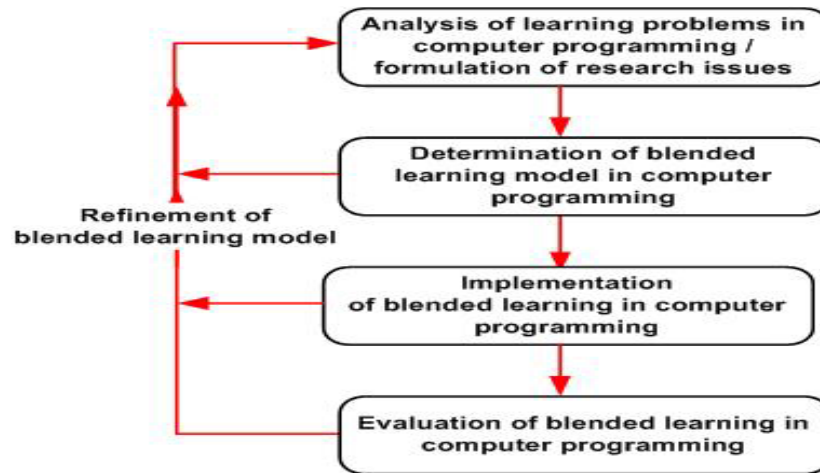
## Design-Based Research Approach

Among many approaches to technology-enhanced learning environments, Design-Based Research is one of the most appropriate approaches for designing and evaluating blended learning through successive cycles of refinements (Barab & Squire, 2004; The Design-Based Research Collective 2003; Terashima, 2004). Design-Based Research embodies specific theoretical claims about teaching and learning, and helps to understand the relationships among learning theory, information technology, and educational practice. Design and research are not isolated as in traditional instructional design and research. There are interdependent and reciprocal (Wang & Hannafin, 2003). The essential characteristic of Design-Based Research is that it describes a continuous cycle, or feedback loop, of gradual refinement of the learning model. Design-Based Research is suitable for blended learning, which needs to evolve rapidly in order to ensure the relevance, appropriateness, and effectiveness of the learning model. Thus, a continuous evolution is of paramount importance for the quality of blended learning. When applied to blended learning in computer programming, Design-Based Research is conducted in four phases: analysis, design, implementation, and evaluation (Figure 1):

1. Design-Based Research begins with the *analysis of the problems and deficiencies of current educational practice* in computer programming, the *formulation of hypotheses and research questions* of interest, and the review of relevant literature.
2. It continues with the *determination of the blended learning model* that will be used to solve learning problems of current educational practice. The model supports the designers' work, forming the foundation for evaluation and research.
3. The implementation phase is concerned with the *application of the blended learning model* to computer programming using multiple methods for collecting empirical data, e.g. survey questionnaires, interviews, observations, etc.
4. The evaluation phase is concerned with the *evaluation of the blended learning model* through the systematic analysis of the data collected, critical evaluation, and reflection.

Analysis, design, implementation, and evaluation are interdependent and reciprocal (Wang & Hannafin, 2003). Refinements are continually made through successive cycles of experimenta-

tions, where the shortcomings of each cycle are identified, redesigned, re-implemented, and re-evaluated.



**Figure 1: Design-Based Research as a feedback loop with four major phases**

Through this evolutionary, iterative, and incremental process, it is possible to gain practical insights and generate pragmatic design principles that help teachers and educational researchers to improve blended learning and its theoretical foundation in computer programming.

## Research Issues

The underlying hypothesis of this work is that blended learning has the potential to improving the learning of computer programming. To test this hypothesis, the application of the model should not be limited to a single preliminary experiment since possible learning benefits can be achieved only through the iterative and continuous cycle of analysis, design, implementation, and evaluation of the learning model in varied contexts. Hence, the objectives of this preliminary study are four research issues:

1. To analyze learning problems and deficiencies associated with current pedagogical practices in introductory computer programming.
2. To design a blended learning model that blends face-to-face learning and online learning. The model will be used to solve learning problems of current educational practice in introductory computer programming.
3. To report on and evaluate an implementation of the model in Java programming, and to find out whether novice students, without prior knowledge in programming, are able to acquire basic Java programming concepts and to solve programming tasks using the learning model.
4. To discuss the implications of the learning model and to identify critical factors of success when applying the model to introductory computer programming.

## Learning Problems and Deficiencies in Introductory Computer Programming: A Literature Review

Programming is a difficult matter because it is more a skill than a body of knowledge. According to Pollack and Schertz (2003), the most common approach to programming among novice students is that of “*bricolage*”: Students develop programs directly on the computer, and tend to skip

the phases of analysis and design. They develop their programs gradually by testing them on various examples of input. Novice students practicing “*bricolage*” are incapable of explaining and justifying their algorithms (Ben-David Kolikant & Pollack, 2004).

In line with this practice, novice programmers come up with mental or cognitive obstacles and misconceptions about computing that make it difficult to understand the functioning of programs or the construction of algorithms. Misconceptions can be attributed to the fact that students possibly interpret computer programming as communication among humans (Dagdilelis, Satratzemi & Evangelidis, 2004). According to (Ben-Ari, 2001), misconceptions are hard to change, unless students acquire a model of a computer.

For solving learning difficulties, educators tried to apply learning theories to computer programming, in particular the constructivist learning theory (Ben-Ari, 2001; Exton, 2002; Gibbs, 2000; Gonzales, 2004; Hadjerrouit, 1999; Lui, Kwan, Poon, & Cheung, 2004; Mead, Gray, Hamer, James, Sorva, et al., 2006; Sajaniemi & Kuittinen, 2005; Wulf, 2005). They argue that novice students must construct a valid model of a computer in order to deal with the difficulties of learning programming. Moreover, proficiency in programming requires the acquisition of higher-order thinking skills, such as analysis, design, analogical thinking, reuse, evaluation, and reflection. Currently, however, only few educators systematically apply constructivism to computer science (Berglund, Daniels & Pears, 2006), and constructivist learning strategies are only beginning to emerge.

Another solution to the learning of programming is to use information technologies, e.g. online programming systems, Web-based programming tutors, online learning systems, or similar software, that give appropriate feedback while working on programming assignments. Currently, however, there are few examples of application of online learning and Web technologies within computer programming. Hence, it is not possible to draw general conclusions about the effect of online learning systems and similar software (Clancy et al, 2003; Conolly & Stansfield, 2007; Hadjerrouit, 2005; Schwieren, Vossen & Westerkamp, 2006; Shaffer, 2005). In addition, most applications focus on technological aspects rather than pedagogical principles based on learning theories, which is not sufficient for helping students acquire programming skills.

Finally, experiences with blended learning models, which blend face-to-face learning and Web-based systems or similar software, are increasingly becoming an attractive option as new innovative technologies become increasingly available (Bonk & Graham, 2006). However, combining face-to-face learning with innovative technologies cannot provide effective teaching and efficient solutions for learning, unless blended learning is designed in conjunction with effective learning strategies and approaches (Luca, 2006). Currently, however, blended learning solutions to the programming problem are still in their infancy (Doderio, Fernandez & Sanz, 2003). This is not sufficient to draw general conclusions about the effect of blended learning models on computer programming.

As a result, even if some progress has been made in solving some learning problems using learning theory and information technologies, the difficulty of learning introductory programming for novice students still remains. Some educators believe that solving the learning problems associated with introductory computer programming requires a radical change from traditional instructional methods, e.g. behaviorism, to constructivist learning environments (Wulf, 2005), mostly because programming is considered as a skill that students need to acquire through an active construction process. Furthermore, according to McDougali and Boyle (2004), programming is an inherently social activity as good programs are developed not in isolation; instead they involve interaction with other people. Programming skills and techniques are acquired from a wide variety of sources; many of them are not classroom-based.

## Blended Learning: A Definition

What is blended learning? Clearly, there is no clear and unequivocal definition of the concept of blended learning. Definitions in the research literature are partially exclusive and sometimes contradictory, and there are few common terms used consistently (Anohina, 2005; Nocols, 2003). It is indeed difficult to distinguish the term “blended learning” from other terms such as “virtual learning”, “technology-based learning”, “distance learning”, “open and flexible learning”, “network learning”, “online learning”, “multimedia-based learning”, “Web-enhanced learning”, “Internet-enabled learning”, and similar terms. Some researchers define the term so broadly that would be hard to find any learning system that is not blended. Thus, there is a wide variety of responses to blended learning, but most of definitions are just variations of few common terms. According to (Bonk & Graham, 2006) the most commonly answers are:

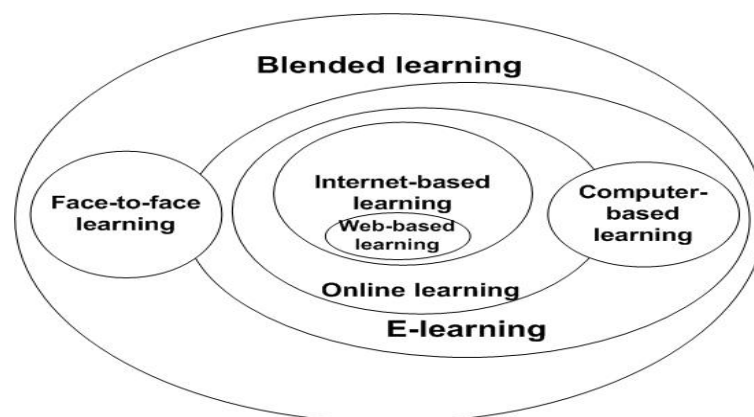
1. Combining instructional modalities or delivery media and technologies (traditional distance education, Internet, Web, CD ROM, video/audio, any other electronic medium, e-mail, online books, etc.)
2. Combining instructional methods, learning theories, and pedagogical dimensions
3. Combining e-Learning and face-to-face learning

The third definition is the working definition of this work. It is a broader definition than the two first positions except that the delivery technology is computer-based. It includes the first and the second definitions with some important modifications:

1. Both face-to-face learning and e-Learning incorporate a combination of learning theories and pedagogical strategies
2. The instructional modality or delivery mode of e-Learning is exclusively computer-based

According to Bonk and Graham (2006) this definition reflects that blended learning “*is the combination of instruction from two historically separate models of teaching and learning: Traditional learning systems and distributed learning systems. It emphasizes the central role of computer-based technologies in blended learning.*”(p. 5).

To sum up, blended learning is a combination of e-Learning and face-to-face learning (Figure 2). E-learning itself includes both network-based (online learning, Internet-based learning, and Web-based learning) and non-network-based learning (computer-based learning).



**Figure 2: Components of blended learning**

## Pedagogical Foundations of Blended Learning

Important to the design of blended learning is a pedagogical foundation built on solid learning theory. Literature reviews suggest that learning theories can be related to three widespread models: cognitivist, constructivist, and socially situated model of learning. The remainder of this section describes the most important characteristics of the learning theories and presents a three-stage model - the learning cycle - that retains the features of each one.

### **Learning Paradigms**

The *cognitive learning paradigm* began a shift from behaviorist practices, which emphasized external behavior, to a concern with the internal mental processes of the mind and how they could be utilized in promoting effective learning. The cognitive approach emphasizes the learner's schema as an organized knowledge structure (Bruner, 1990; Gagne et al, 1993). Unlike behaviorism, cognitivism recognizes that the human mind is not simply a passive recipient of knowledge. Rather, the learner interprets knowledge and gives meaning to it. New knowledge is integrated with prior knowledge. The cognitive perspective of learning refers to mental activity, such as analytical reasoning and critical thinking. When teachers apply a cognitive approach, they focus on the understanding of concepts and their relationships. If learners are able to understand the connections between concepts, break down information and rebuild it with logical connections, then their understanding will increase.

The *constructivist paradigm* of learning views knowledge as a constructed entity made by each and every learner through a learning process. Constructivism frames learning less as the product of passive transmission than a process of active construction whereby the learners construct their own knowledge based upon prior knowledge and experience (Duffy, Lowyck & Jonassen, 1993; Piaget, 1971; Steffe & Gale, 1995). Constructivist learning requires learners to demonstrate their skills by constructing their own knowledge when solving real-world problems. Therefore, the constructivist model calls for learner-centered instruction because learners are assumed to learn better when they are forced to explore and discover things themselves. Constructivism includes task-based activities and project-based learning within ill-structured domains and authentic tasks taken from the real-world.

This *socially situated learning paradigm* can be seen as a correction to constructivism, in which learning is disconnected from the social context. Whereas in the constructivist paradigm learning is assumed to occur as an individual learner interacts with study material, this perspective regards learning as socially situated and knowledge as socially distributed (Vygotsky, 1978; Wengler, 1998). Learning occurs as learners exercise, test, and improve their knowledge through discussion, dialogue, communication, collaboration, information sharing, and interaction with others. Vygotsky (1978) argued that the way learners construct knowledge, think, reason, and reflect on is uniquely shaped by their relationships with others. He argued that the guidance given by more capable others, allows the learner to engage in levels of activity that could not be managed alone.

### **The Learning Cycle**

The literature on learning theories points to the fundamental philosophical differences between them (Lin & Hsieh, 2001). However, in practice, a blend of learning theories is being used. Indeed, instructional designers tend to believe that what works in a learning situation is a subtle combination of learning theories. Hence, instructional designers must allow circumstances surrounding the learning situation to help them decide which approach to learning is most appropriate. It is necessary to realize that some learning problems require prescriptive solutions, whereas other are more suited to cognitive, constructivist, or situated learning (Karagiorgi & Symeou, 2005).

Along the same line of argument, Mayes and Fowler (1999) proposed a three-stage model or learning cycle, in which they identified three types of learning – conceptualization, construction, and dialogue. The essential characteristic of the learning cycle is that it describes a continuous cycle, or feedback loop, of gradual refinement of understanding. Accordingly, learning develops in three phases, beginning with conceptualization, progressing through construction to dialogue. Conceptualization is characterized by the process of interaction between the learners’ pre-existing framework and teacher’s knowledge. The construction phase – the intermediate phase of learning – refers to the process of building and combining concepts through their use in the performance of meaningful tasks. The dialogue phase refers to the testing of conceptualizations and the creation of new concepts during conversation with both fellow learners and teachers. Dialogue emerges through collaborative learning.

The three stages of the learning cycle include elements that are closely related to learning theories. Conceptualization is associated with the cognitive learning theory as it focuses on concepts and their relationships. The construction phase is related to the constructivist learning theory as it aims at the construction of new knowledge and its use in the performance of task-based activities. The dialogue phase is based on the socially situated learning theory as it is concerned with dialogue, group collaboration, and discussion.

As a result, at any stage of the learning cycle there needs to be an emphasis on the three stages of the learning cycle: learning as cognition, learning as knowledge construction, and learning as dialogue and social practice.

### ***Information Technologies Used in the Learning Cycle***

Information technologies can be used to implement the learning cycle. Mayes and Fowler (1999) characterized the types of information technologies used to achieve each stage of the learning cycle as primary, secondary, and tertiary courseware:

- *Primary courseware* is intended mainly to present the concepts of the subject matter.
- *Secondary courseware* focuses on the set of software tools that support the performance of task-based activities.
- *Tertiary courseware* is the learning material which has been produced by previous learners. Additionally, it may consist of online dialogues between learners and teachers, as well as online group discussions and collaborations.

Mayes and Fowler’s model has been adapted by Roberts (2003) to categorize three uses of the Web. Similarly, the model can be adapted to categorize three uses of blended learning.

### ***Mapping the Learning Cycle onto Blended Learning***

The learning cycle can be adapted to specify the blended learning model. The mapping of the learning cycle onto a blended learning model results in blending at three different levels (Figure 3):

- *Blending at the conceptualization phase.* Blending at this level occurs when the learning combines face-to-face learning with primary courseware. In this phase, the student is acquiring knowledge.
- *Blending at the construction phase.* Blending at this level occurs when the model combines learning activities with secondary courseware, e.g. online task-based activities. In this phase, the learner is involved in constructing new knowledge.
- *Blending at the dialogue phase.* Blending at this level occurs when the learning model combines face-to-face dialogue with tertiary courseware, e.g. online discussion and group collaboration.

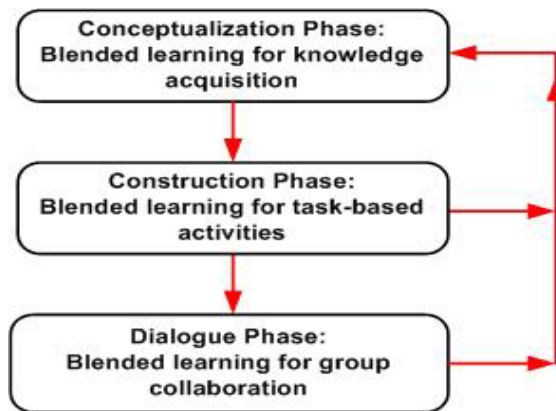


Figure 3: Mayes and Fowlers’ learning cycle as feedback loop adapted to blended learning

## Applying the Blended Learning Model to Java Programming: A Case Study

A case study approach was used to explore the application of the blended learning model to Java programming. The goal of the study is to provide an understanding of how the model can facilitate the students’ learning of programming. In addition, the study shows how the blended learning model evolved through design, evaluation, redesign, and reevaluation during the fall semester of 2006.

Java programming is taught during the first semester of the Bachelor Study Program in informatics at the Faculty of Mathematics (Hadjerrouit, 2006). The primary objective of the course is to help the students to gain practical programming experiences through involvement in Java programming activities.

The course was given in the form of two hours of lectures and four hours of laboratory work per week during a 15-week semester. As shown in Table 1, the course content was divided into 8 blocks (Hadjerrouit, 2006). Block 1 introduced the basic principles of Java programming, as well as variables, assignments, data types, expressions, and operations. Block 2 explained the basic concepts of input and output. Block 3 focused on control statements: *if*, *if-else*, *switch*, *while*, *do-while*, and *for*. Block 4 covered array structures. Block 5 focused on strings. Block 6 explained Applets. Block 7 presented methods. Finally, block 8 was devoted to repetition of the material covered during the semester, and to solutions of programming assignments and past exams. Each block was designed to promote the blended learning model in three phases: conceptualization, construction, and dialogue.

Timeframe	Block	Focus
Week 1	Block 1	Principles of Java programming. A first Java Program
Week 2		Variables, assignments, data types, expressions, and operations
Week 3		Variables, assignments, data types, expressions, and operations
Week 4	Block 2	Basic principles of input / output, easyIO package, input/output operations
Week 5		Basic principles of input / output, easyIO package, input/output operations
Week 6	Block 3	Control statements I: <i>if</i> -statements, <i>if-else</i> -statements, nested <i>if</i> -statements, <i>switch</i> -statements
Week 7		Control statements II: <i>while</i> -loop, <i>do-while</i> -loop

Week 8		Control structures III: <i>for</i> -loop, nested <i>for</i> -loop
Week 9	Block 4	Arrays I: Declaring array variables, copying arrays, multidimensional arrays, searching, and sorting arrays.
Week 10		Arrays II: Declaring array variables, copying arrays, multidimensional arrays, searching, and sorting arrays
Week 11	Block 5	Strings: Text strings and basic operations, concatenation, and translations
Week 12	Block 6	Applets: Sample Applets, executing Applets in the <i>applet-viewer</i> and in the Web browser, Applet life cycle
Week 13	Block 7	Methods: Creating and calling methods, passing parameters, overloading methods, scope of local variables
Week 14	Block 8	Repetition: Solutions of programming assignments and past exams
Week 15		Repetition: Solutions of programming assignments and past exams

### **Conceptualization Phase: Java Concepts**

The course was designed to support the conceptualization phase of the blended learning model, that is to say the process of interaction between the students' pre-existing knowledge structures and the key concepts of the subject matter. At any time, it is quite important for the teacher to be aware of the current state of the students' pre-existing knowledge in programming, skill level, and misconceptions about programming. Accordingly, the teacher needs to prove the knowledge that the students previously constructed and evaluate whether this knowledge conflicts with the knowledge being taught. For instance, if the new concept to be taught is the *while-loop* and students' prior knowledge are *if-statements*, *integers*, and *instruction sequences*, then they should be able to construct the concept of while-loop using their prior knowledge. The role of the teacher is to demonstrate that using their prior knowledge for constructing the while-loop is not an adequate way for solving problems related to while-loops. Using comparisons and conflict strategies, the teacher should be able to demonstrate the adequacy of the new concept. An advantage of programming is that a number of advanced programming concepts can be addressed with less advanced ones (Hadjerrouit, 1999). For example, calling a method three times is similar to writing three times the same sequence of instructions. Thus, the teaching strategy here consists of connecting the new concept with students' prior knowledge. There are a number of strategies that can be used to construct Java concepts:

- a) Use *multiple representations* of the Java concept to be learned. A representation may be linguistic, verbal, symbolic, or pictorial. A precise definition of any Java concept requires many representations and the translation from one to another.
- b) *Compare and contrast*. Organize Java concepts in terms of similarities and differences.
- c) *Make forward and backward references* in order to connect students' prior programming knowledge and new Java concepts.
- d) *Explore extended contexts* of applicability of Java concepts.
- e) *Classify and categorize* Java concepts in terms of their common features.

### **Construction Phase: Java Programming Activities**

The course was designed to support the construction phase of the learning model, that is to say the process of building new conceptualizations through the performance of task-based activities. The process of constructing new Java programs is the product of previous Java program constructions. Thus the process is recursive and should be organized in a spiral manner so that students continually build upon what they have already learned. Important for the programming construc-

tion process is the design of authentic task-based activities that are intrinsically motivating (Hadjerrouit, 1999). The activities should be designed to “anchor” new concepts to previous ones by working on representative problems. Accordingly, there are a number of strategies for constructing Java programs:

- a) *Analyze and design*. These activities are important because students are inclined to skip the analysis and design steps and go directly to coding the program. Therefore, students should learn to analyze the programming problem, break down the problem-solving process into its main components and design an appropriate algorithm before coding. Clearly, students need to develop these necessary skills before coding.
- b) *Reuse of previous solutions*. This strategy is based on the idea that solutions of previous problems can be reused with some modifications to solve similar problems. To be able to reuse similar solutions students need to acquire analogical thinking skills.
- c) *Study experts’ programming solutions* from textbooks, and reuse them with slight modifications. This activity requires analogical thinking skills.
- d) *Compare and contrast* alternative solutions to find the most efficient one.
- e) *Predict the behavior of the program*. It is critical that students make predictions – right or wrong – beforehand, so that they can anticipate the program’s behavior.
- f) *Generate multiple solutions*. Students usually have only one way of programming a solution. Once they have found a solution, they often begin to execute it immediately and don’t stop either to consider its applicability or find other ways of solving the problem. However, efficient programming cannot occur unless students choose from a set of valid solution paths.

### ***Dialogue Phase: Interactions, Collaborations, and Discussions***

Finally, the course was designed to support the dialogue phase of the blended learning model, that is to say the testing of students’ understanding of Java concepts and programming activities during dialogue. This phase can be performed separately or in parallel with the first and second phase of the learning model. There are a number of strategies for implementing this phase (Hadjerrouit, 1999):

- a) *Explain* (summarize, describe, discuss). Explaining the programming solution process gives students practice in applying their ideas and solutions to new situations.
- b) *Reflect on* (evaluate, integrate, extend, generalize). After programming activities, students benefit from reflecting on what they have just done. Evaluating the programming solution helps to reflect on the construction process.
- c) *Meta-communication*. Too often, students conceive a solution is just a program that works for them, rather than a program that is readable and understandable for others. As a result, students tend not be very precise when communicating the results of their programming activities. Therefore, precision in meta-communication is essential for successful programming. Teachers must discuss why precision is important and help students being precise. This activity is closely related to the “explain” and “reflection” activities.

### ***Online Resources***

Likewise, the online resources of the blended learning model were designed to promote the learning cycle in three phases: conceptualization, construction, and dialogue, according to the principles described in section “Information Technologies Used in the Learning Cycle”. The platform of the online resources was a Web-based Learning Management System (Hadjerrouit, 2006).

To support the conceptualization phase, the online resources were designed as *primary courseware* to present the subject matter, enabling the access to resources that offer various types of information that can be used to either a greater understanding of Java concepts, or to obtain further information about them. The most important criteria for designing the online resources for conceptualization were a well-structured presentation, easy accessibility, and powerful explanation of the information in order to effectively transmit knowledge to the students. Hence, the design strategies and associated online resources for the conceptualization phase were:

- Break down Java knowledge into concepts that can be constructed using the pedagogical strategies described above.
- Provide a well-structured online presentation and organization of the Java concepts.
- Provide easy and user-friendly accessibility of the concepts and links to related study material.
- Provide a well-structured description of the Java concepts using a clear and understandable language.

Then, the online resources were designed to support the construction phase, that is to say the process of constructing and performing programming tasks. In order to perform programming tasks, students should have access to resources that support active, independent, and self-reflective learning. Thus, the online resources were designed as *secondary courseware*. The resources required the design of programming tasks - rather than the presentation of Java concepts - in order to encourage students to construct Java programs. Thus, the most important online resources for the construction phase were:

- Online well-designed programming examples that students may follow when they perform programming activities.
- Online presentation of teachers' programming solutions that students may adapt and reuse with some modifications to solve new programming problems.
- Online reusable Java code that can be adapted, modified, and extended to meet the requirements of new programming tasks.
- Multiple representation of the online information using various elements, such as text, graphics, pictures, symbols, etc.
- Links to online programming exercises and past exams.

Finally, the course was designed to support the dialogue phase of the blended learning model, enabling students to discuss their programming solutions, through e-mail and eventually LMS-enabled discussions with the teacher and fellow students. Thus, the online resources were designed as *tertiary courseware* to support collaborative learning. Tertiary courseware included:

- Synchronous communication (chats), in which students could communicate with each other simultaneously in real-time.
- Asynchronous communication (e-mail, discussion forum, electronic messages), in which students are separated by time and space.
- Students' submissions of programming solutions, individually or as a group. The teacher can comment, grade the solutions, and give feedback.

## ***The Blended Learning Model in Use***

Each week of a 15 week-semester course the teacher decides in advance the concept(s) to be taught and the underlying programming activities to be performed. The activities are formulated in such a way to connect the new concept(s) to be taught to previous ones by working on representative and motivating problems. The objective of classroom teaching is to generate understanding of Java concepts. By using situated examples, the teacher enables the students to understand Java concepts.

Then, the students try to construct solutions to programming problems. For example, if the Java concept to be constructed is the while-loop, then the students would do activities that are related to while-loops. Students work individually or in small groups. The role of the teacher is to act as a guide and facilitator of learning by directing the students' thinking in the right direction. The programming process itself is an iterative process over 15 weeks with continuous refinements, revisions, modifications, and reuse of previous solutions and examples that were constructed before. Normally, students spend a considerable amount of time performing programming activities. In addition, they have to explain, reflect on, and summarize the solution process to the teacher.

Finally, discussions with the students allow the teacher to synthesize the new Java concept(s) and students' programming activities, and eventually correct the students' solutions by providing new or supplementary information. Students get the opportunity to raise questions regarding the specific activity or more general problems. In this phase, the teacher can provide supplementary information and discuss how the program can be re-used in similar situations.

## **Evaluation Methods**

The goal of the evaluation was to test the research hypothesis as to whether the blended learning model improved the students' learning of Java programming. According to the Design-Based Research paradigm, it may be necessary to continually refine the model through successive cycles of experimentations, where the shortcomings of each cycle are identified, redesigned, re-implemented, and reevaluated. The number of experiments depends on the course duration, which was three months in this case study. As a result, two cycles of experimentations and evaluations were performed during this short timeframe. New cycles of experimentations and evaluations are planned in future courses.

## ***Participants***

Data from the present case study came from survey questionnaires completed by 11 students that were registered for the course in the fall semester of 2006.

## ***Data Collection Methods***

Three survey questionnaires were employed to investigate the students' learning:

- A first pre-questionnaire was provided to the students before the first programming activity began. This was designed to collect data related to students' prior knowledge, experience, and skill level in computer programming.
- A second pre-questionnaire was provided to the students after one month of Java programming. This was designed to collect data related to the level of difficulty of the subject matter, online resources, teacher's feedback, and students' overall satisfaction with the course.
- A post-questionnaire was delivered to the students one week before the end of the course. Both technical and pedagogical issues were addressed in this questionnaire.

In addition to the survey questionnaires, attention was devoted to the following supplementary data collection methods in order to strengthen the validity of the evaluation:

- a) Exam scores achieved by the students.
- b) Informal dialogues and discussions with the students.
- c) Teacher's observations over a three-month time period.
- d) Comparing the data from 2006 with data collected in the fall semester of 2004.
- e) When possible, finding evidence in the research literature that supports the data collected.

The method used for data analysis consisted of finding diverse pieces of evidence from four different perspectives: the teacher's perspective, the students' perspective, the perspective of the research literature, and the perspective of the data collected in the fall semester of 2004.

To ensure the rigor and validity of analysis, data sources were triangulated through overlapping of diverse pieces of evidence and perspectives (Bryman, 2004), and an active search for conforming and disconfirming evidence was made through new informal discussions with the students and teacher's observations. The goal was to achieve a deeper understanding of the students' learning experiences.

### **Evaluation Measures**

To measure the students' responses to the post-evaluation, a five-point Likert scale from 1 to 5 was used, where 1 was coded as the lowest and 5 as the highest (1 = "Strongly Disagree"; 2 = "Disagree"; 3 = "Neither Agree or Disagree"; 4 = "Agree"; 5 = "Strongly Agree"). The students were asked to respond to the evaluation issues by placing a cross "X" in the appropriate box using the scale provided (five-point Likert scale). Total responses are represented as numbers.

## **Evaluation Findings**

As mentioned above, two cycles of experimentations and evaluations of the blended learning model were performed within the timeframe of the course. Hence, evaluation findings are described with regard to:

- The *pre-evaluation* that was performed one month after the beginning of the course, and the *redesign of the learning model* according to the findings of the pre-evaluation and students' recommendations.
- The *post-evaluation* that was performed one week before the end of the course.

### **Pre-evaluation**

Before the pre-evaluation, a pre-questionnaire was used to collect data related to students' prior knowledge, experience, and skill level in computer programming. From the collected data, it was obvious that most students did not have any knowledge background in computer programming.

The pre-evaluation was conducted one month after the beginning of the course and was concerned with collecting data according to the three-stage model of the blended learning model: conceptualization, construction, and dialogue. Accordingly, the following issues were evaluated:

- a) The process of interaction between students' pre-existing knowledge and the level of difficulty, scope, and depth of the concepts presented during the conceptualization phase.
- b) The degree of support provided by the construction phase of the blended learning model to complete programming tasks.

- c) The extent to which the dialogue phase of the blended learning model supported dialogue among students and teacher.
- d) The degree of support provided by the online resources to the understanding of Java concepts and Java programming activities.

The pre-evaluation results were as follows:

- A number of students found that the level of difficulty, scope, and depth of the subject matter compared to the students' background knowledge is relatively high. This was not surprising given the fact that most students did not possess sufficient prerequisite knowledge in programming, and that the subject matter itself is quite difficult for novice students, according to the research literature.
- Some students think that Java programming is a challenging task. They recommended more teacher guidance and feedback, as well as pedagogically sound explanations. However, even if the learning of programming was difficult, most students found that the degree of technical support provided by the online resources was very good whenever they needed access to course information at any time and any place.
- While the majority of the students believed that the electronic platform contain useful online resources - e.g. programming examples and associated solutions, that be can be executed by a group of students in online collaboration - most students preferred face-to-face discussions, which were considered to be highly important to the learning of Java programming. As a result, few students used tertiary courseware.

### ***Redesigning the Learning Model***

The pre-evaluation enabled students to suggest improvements to the remainder of the course. Furthermore, the findings of the pre-evaluation gave the teacher the opportunity to redesign the learning model. The goal of redesigning the model was minimizing the students' work load resulting from the difficulty, scope, and depth of the subject matter in order to help them to concentrate on what really matters: the learning of Java programming and acquisition of programming skills. Hence, changes were made as follows:

- First, course material that was difficult for the students and not very important to the learning process was partly removed from the online resources.
- Second, substantial attention was devoted to teacher's guidance, pedagogically sound explanations of programming activities, and appropriate feedback.
- Third, the teacher considered how to improve face-to-face collaboration and supervision both in the classroom and computer lab, where students performed their programming activities.

### ***Post-evaluation***

To goal of the post-evaluation was to assess the students' learning after the redesign of the learning model and associated online resources. The teacher used a survey questionnaire that was delivered to the students one week before the end of the course. This questionnaire contained more issues than the one used in the pre-evaluation. The questionnaire consisted of 37 questions and issues. It addressed both technical and pedagogical issues:

- Technical evaluation issues addressed the extent to which the online resources provided support to the learning of Java concepts and programming.

- Pedagogical evaluation issues addressed the extent to which the blended learning model provided support to the learning process.

Technical and pedagogical evaluation procedures are closely related to each other (Melis, Weber & Andres, 2003; Nokelainen, 2004). The goal should be minimizing the learners' work resulting from the interaction with the online resources in order to free more resources for the learning process itself.

The evaluation of technical issues is a self-evident requirement, but it is not sufficient for evaluating the pedagogy of the blended learning model. Hence, evaluation procedures must be extended to capture pedagogical issues that are fundamental to learning. The criteria that influence the pedagogical evaluation are those that are associated with the learning model. Hence, the starting point for defining pedagogical evaluation issues was to split the learning process into three types of learning with respect to the learning model: a conceptualization phase, a construction phase, and a dialogue phase.

### ***Students' Perceptions of the Online Resources***

The evaluation of technical issues is a widely used method to assess online learning (Agostinho & Herrington, 2004; Nielsen, 2000; Shiratuddin, Hassan & Landoni, 2003; Storey, Phillips, Maczewski & Wang, 2002).

The following evaluation addressed the extent to which the online resources provided technical support to Java programming. The main objectives of this evaluation were the following 7 issues:

1. The content of the online resources is concise and well-organized.
2. The navigation is straightforward and intuitive.
3. The online resources are well-designed, user-friendly, and easy to access.
4. The online resources are structured in a clear and understandable manner.
5. I recommend the reuse of the online resources in future courses.
6. I am globally satisfied with the online resources.
7. The online resources should be improved.

Findings from the technical evaluation are shown in Table 2. Total responses are represented as numbers.

The evaluation of the technical issues of the online resources shows that the average score was 4.13. More specifically, the evaluation shows the following trends:

As Table 2 indicates, most students were globally satisfied with the online resources. All students (100%) strongly agreed or agreed that they are satisfied with issue 1, 2, 3, 4, and 6 of the online resources (content, navigation, usability, structure, global satisfaction). Moreover, 10 students (90.90 %) recommended to reuse the online resources in future courses (issue 5). In addition, 63.63 % of the students disagreed or strongly disagreed that the online resources should be improved (issue 7). From the results it can be inferred that the online resources were well implemented. These responses were positively correlated with the students' high level of satisfaction with the online resources.

<b>Table 2. Evaluation of online resources</b>					
<b>Evaluation issues</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neither Agree or Disagree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1. The content of the online resources is concise and well-organized.	6	5	0	0	0
2. The navigation is straightforward and intuitive.	4	7	0	0	0
3. The online resources are well-designed, user-friendly, and easy to access.	6	5	0	0	0
4. The online resources are structured in a clear and understandable manner.	3	8	0	0	0
5. I recommend the reuse of the online resources in future courses.	5	5	1	0	0
6. I am globally satisfied with the online resources.	5	6	0	0	0
7. The online resources should be improved.	1	0	3	5	2

In addition to the issues of the survey questionnaire, the students were asked to answer three additional questions (Tables 3, 4, and 5). Responses to these questions are presented below.

<b>Table 3. Help provided by the online resources</b>	
<b>How much the online resources helped you to learn Java?</b>	<b>Frequency</b>
1. Very useful	5
2. Useful	6
3. Not very useful	0
4. Didn't use them	0

In total, all students believed that the online resources are very useful (45.45 %) or useful (54.54%) for learning Java concepts and programming.

<b>Table 4. Usefulness of online resources</b>	
<b>On reflection, how useful were the online resources as a learning resource?</b>	<b>Frequency</b>
1. Very useful	5
2. Useful	5
3. Not very useful	1
4. Didn't use them	0

In total, most students think that the online resources are very useful (45.45%) or useful (45.45%) as a learning resource. Only one student didn't believe that the resources are useful.

<b>How did you use the online resources?</b>	<b>Frequency</b>
1. I used all the relevant learning resources each week	6
2. Used the ones needed to help with programming tasks	7
3. I read them quickly and returned to the ones needed later when I had more time	5
4. I used them for revision	3
5. I used them for the exam	2

In total, 54.54 % of the students used all the relevant learning resources each week. In addition, 63.63 % used the ones needed to help with programming tasks, and 45.45 % skimmed through them and returned to the ones needed later when they had more time. Three students (27.27 %) used them for revision, and two (18.18 %) for the exam.

As a result, it appears that responses to the additional questions are in correlation with the students' high level of satisfaction with the online resources. Hence, it can be inferred that the online resources were very useful or useful for the learning of Java concepts and programming.

### ***Students' Perceptions of Pedagogical Issues***

The evaluation of pedagogical issues addressed the extent to which the blended learning model provided support to the learning process. Pedagogical issues were evaluated from the perspective of the blended learning model and associated learning cycle with three phases: conceptualization, construction, and dialogue.

#### **Conceptualization phase**

This phase included 9 issues. The results are displayed in the Table 6. Total responses are represented as numbers.

The evaluation of the conceptualization phase shows that the average score was 3.86. More specifically, the evaluation shows the following trends:

- Overall satisfaction with the course (Issue 1): 90.90 % of the students strongly agree or agree.
- Understanding of Java programming (Issue 2): All students (100 %) strongly agree or agree.
- Appropriateness of support to programming tasks (Issue 3): 72.72 % strongly agree or agree.
- Student engagement and motivation (Issue 4): 54.54 % of the students strongly agree or agree.
- Motivating programming tasks (Issue 5): 63.63 % of the students strongly agree or agree.
- Level of difficulty (Issue 6): 72.72 % of the students strongly agree or agree.
- Alignment with learning objectives (issue 7): 54.54 % of the students strongly agree or agree.

- Consolidation of knowledge throughout the course (Issue 8): 100 % strongly agree or agree.
- Appropriateness of course textbook (Issue 9): 72.72 % of the students agree.

<b>Evaluation issues</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neither Agree or Disagree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1. Overall, I am satisfied with the course.	2	8	1	0	0
2. The course helped me to construct my own knowledge and understanding of Java programming.	4	7	0	0	0
3. The course provided appropriate support to help me complete programming tasks.	2	6	3	0	0
4. The course supported my motivation and engagement in learning Java programming.	1	5	5	0	0
5. The course provided motivating programming tasks.	1	6	4	0	0
6. The knowledge demands and the level of difficulty of the course are appropriate.	1	7	3	0	0
7. The course content is closely aligned with the intended course objectives and goals.	2	4	5	0	0
8. Programming tasks enabled me to reflect on and consolidate my learning of programming at various stages throughout the semester.	4	7	0	0	0
9. The course textbook helped me to learn programming and to solve programming tasks.	0	8	3	0	0

Analysis of the responses to the categories "Strongly Agree" and "Agree" shows that the students were globally positive about the evaluated issues. Considering, in addition, that some of the neutral responses ("Neither Agree or Disagree") can be regarded as a tacit approval of the category "Agree", otherwise some of the students would have chosen the category "Disagree", one can be satisfied with the evaluation results. Hence, it can be inferred from the students' responses to the questionnaire that the conceptualization phase was globally well implemented as it provided good support to the understanding of Java concepts and performance of programming tasks. The results also show that there is a positive correlation with the evaluation of the online resources.

### **Construction phase**

The construction phase of the learning process included the following 11 issues (Table 7). The evaluation was concerned with the students' individual programming construction process and the acquisition of higher-order skills, such as analysis, design and reuse skills, making connections between concepts, explaining, justifying, making predictions, developing alternative solutions, reflecting on and evaluating the solution process, etc.

<b>Evaluation issues</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neither Agree nor Disagree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1. I analyze programming tasks before programming them.	0	3	5	3	0
2. I design an algorithm for programming tasks before programming them.	0	1	2	7	1
3. I make connections to previous programming solutions when performing programming tasks.	2	6	2	1	0
4. I make connections to my own previous programming solutions when doing programming tasks.	5	6	0	0	0
5. I reuse programming knowledge and solutions when performing programming tasks.	3	3	4	1	0
6. I develop alternative (multiple) solutions to programming tasks.	0	1	5	3	2
7. I compare and contrast the solutions to find out the most efficient one.	1	3	1	4	2
8. I make predictions about the program's behavior before testing the program solution.	1	2	6	2	0
9. I feel I have a sense of control over my own learning when demonstrating the programming solution process to the teacher.	1	7	3	0	0
10. I test the programming solution with a set of data.	2	8	1	0	0
11. I reflect on and evaluate the programming solution process.	0	4	7	0	0

The evaluation of the construction phase shows that the average score was 3.08. From the point of view of the acquisition of higher-order thinking skills, the evaluation results show the following trends. The results are given in percentage and ordered according to the degree of importance:

- Connecting to previous students' knowledge (issue 4): 100 % strongly agree or agree.
- Program testing (Issue 10): 90.90 % of the students strongly agree or agree.
- Connecting to previous teacher's knowledge (issue 3): 72.72 % strongly agree or agree.
- Communicating the solution process (Issue 9): 72.72 % strongly agree or agree.
- Reusing knowledge (Issue 5): 54.54 % of the students strongly agree or agree.
- Comparing and contrasting solutions (Issue 7): 36.36 % strongly agree or agree.
- Reflecting and evaluating (Issue 11): 36.36% of the students agree.
- Analyzing the problem (Issue 1): 27.27 % of the students agree.
- Making predictions (Issue 8): 27.27 % of the students strongly agree or agree.

- Designing algorithms (Issue 2): 9.09 % of the students agree.
- Developing alternative solutions (Issue 6): 9.09 % of the students agree.

From the results, it appears that the students were very good or good in their efforts to acquire skills related to connecting to previous knowledge and programming solutions, program testing, communicating the solution process, and reusing programming knowledge and solutions (Issue 4, 10, 3, 9, and 5). Otherwise the evaluation indicates lower percentage to the categories “Strongly Agree” or “Agree” with regard to the skills: comparing and contrasting, reflecting and evaluating, analyzing the problem, making predictions, designing algorithms, and developing alternative solutions (Issue 7, 11, 1, 8, 2, and 6).

As a result, it appears there is not a positive correlation neither with the evaluation of the online resources nor with the evaluation the conceptualization phase, except for issue 3 and 4, since there is a clear relation between making connections to previous knowledge and the information available online when performing programming tasks.

### Dialog phase

The dialogue phase included the following 7 issues (Table 8). It was concerned with teacher-student interactions and student-student collaborations. Total responses are represented as numbers.

<b>Evaluation issues</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neither Agree or Disagree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1. The teacher is always well-prepared for giving lectures and supervising programming tasks.	3	7	1	0	0
2. Teacher’s lecturing (PP slides, teaching Java on the blackboard, dialog with students) helped me to understand the key concepts of Java	4	6	1	0	0
3. Discussions and dialog with the teacher helped me identify the strengths and limits of my programming knowledge.	4	3	3	1	0
4. There were a variety of opportunities to interact with the teacher and to ask questions.	2	7	1	1	0
5. The teacher helped me to solve the programming tasks.	5	3	3	0	0
6. Student demonstrations of programming tasks and teacher’s explanations are helpful for understanding Java programming.	4	5	2	0	0
7. I work together and collaborate with other students when solving programming tasks.	6	4	1	0	0

The evaluation of the dialogue phase shows that the average score was 4.15. More specifically, the evaluation shows the following trends:

- Teacher preparation and supervision (Issue 1): 90.90 % strongly agree or agree.
- Teacher lecturing and feedback (Issue 2): 90.90 % strongly agree or agree.
- Discussions with the teacher (Issue 3): 63.63 % strongly agree or agree.
- Interactions with the teacher (Issue 4): 81.81 % strongly agree or agree
- Teacher's mentoring (Issue 5): 72.72 % strongly agree or agree.
- Students' demonstrations & teacher's explanations (Issue 6): 81.81 % strongly agree or agree.
- Collaboration with fellow students (Issue 7): 90.90 % strongly agree or agree.

Summarizing, in some contrast to the construction phase, the dialogue phase was quite good implemented since most students perceived that teacher-student interactions are beneficial to them as they helped them to acquire programming knowledge and perform programming activities. Students felt that they are engaged in their learning of programming when they interact with the teacher. They also benefited from their collaborations with fellow students.

## **Supplementary Evaluation: Exam Scores, Discussions, and Observations**

To strengthen the validity of the evaluation results supplementary evaluation were considered: exam scores, informal discussions with students, and teacher's observations.

Exam scores were based on a six-point scale from A to F, where F was coded as the lowest and A as the highest. Score E was required in order to pass the subject matter. The grades exhibited by the students were as follows: 2 students received an "A", 4 a "B", 2 a "C", and 3 a "D". These grades indicate that the overall exam performance of the students was good.

In addition to exam grades, the author collected data in informal discussions with the students and observations in the classroom and computer lab. To facilitate the analysis of the data, the discussion issues with the students and teacher's observations were closely aligned with those of the survey questionnaires. To ensure the rigor and validity of analysis, an active search for conforming and disconfirming evidence was made through successive informal discussions with the students and observations during the whole semester. The analysis of the data collected in informal discussions with the students and teachers' observations indicated that students were very satisfied with the online resources as these were very useful for grasping programming concepts, solving programming tasks, reusing program code, repetition, and revision. Furthermore, they were globally satisfied with the course content and objectives, teacher's lecturing in classroom, and teacher's guidance and supervision of students' programming activities during the whole semester. However, teacher's observations indicated that many students struggled with the analysis and design phases of the programming process, reflecting on and evaluating their programming solutions, as well as developing alternative solutions. This was particularly visible when they had to think about the programming tasks without using the computer. Clearly, it appears that the students' programming activities were computer-dependent, even if most students were unable to deal with the compilation errors. However, many students could not stay away from the computer despite the teacher's recommendations. As a result, most students wanted a computer-based exam, and not one based on paper and pencil, so that they can test their program code and solutions.

## Conclusions and Discussion

In this section, a summary of findings of the case study and the associated implications for the theory and practice in blended learning environments are presented. The limitations of the study and recommendations for future research are discussed as well.

### ***Summary of Findings***

The findings of this study indicate that the students displayed a high level of satisfaction with the online resources of the blended learning model. This is clearly an improvement compared to the programming course given in the fall semester of 2004 (Hadjerrouit, 2005). Yet, a majority pointed out that they did not take into consideration some of the higher-order thinking skills that are needed in the programming learning process. Finally, most students benefited from interactions with the teacher and collaborations with fellow students.

### ***Implications for the Design and Delivery of Blended Learning***

From the application and evaluation of the blended learning model in Java programming the following implications can be drawn:

First, a key finding of this study is that online resources of the blended learning model were a key factor that positively influenced the students' learning of programming. Overall, the results of the present study indicate that the supporting and facilitating students' online learning might be of great value for the programming learning process.

Second, when online resources are well-designed as a source for subject information, they provide useful support for the conceptualization phase of the blended learning model, enabling the access at any time and any place to course material, programming exercises, past exams and their solutions, and related information that can be used to improve the understanding of Java programming. Despite the difficulty of the subject matter, online resources can provide support to the construction phase of the blended learning model - that is the process of building new knowledge through the performance of programming tasks. Basically, online resources provided support through the adaptation, modification, and reuse of well-designed programming examples and their solutions, and reusable program code available online. Interactions and collaborative learning happened mainly face-to-face and, to some degree, by means of email. The students did not use group discussion forum to engage in dialogue and reflection with fellow learners, because, obviously, face-to-face interactions and collaborations were more important to the learning process than online discussions. Hence, mentoring, coaching, and helping students is not just a matter of online dialogue, it is a human relation as well. Online resources cannot fully replace human dialogue and relationships in the programming process. Thus, many things still need to be done face-to-face, such as providing motivation, helping students with learning difficulties, explaining, discussing, evaluating, reflecting on programming solutions, etc.

Third, it appears that collaborative activities are more important to novice students entering the field of computer programming than the individual acquisition of higher-order thinking skills, such as analyzing the problem, designing an algorithm, comparing and contrasting solutions, developing alternative solutions, making predictions, and evaluating and reflecting on the solution process. It does not mean that the acquisition of these skills is not important to the programming process. It means that the acquisition of these skills is not a necessary prerequisite for learning introductory programming, as long as two conditions are satisfied. First, the teacher provides god help through face-to-face discussion and supervision. Second, students receive a greater amount of guidance in programming activities. But still, students need to acquire analogical thinking and reuse skills, the ability to connect new knowledge to previous knowledge, as well as program testing abilities in order to be able to develop good programming solutions.

Finally, from the above it follows that online resources are highly important but not sufficient to learning programming. Online learning needs to be combined with face-face-learning. This because, contrary to online resources as a source for subject information, which can be designed according to the students' needs, it is more difficult to provide support to the programming process due to the importance of higher-order thinking skills, face-to-face interactions, and collaborations with the teacher and fellow students. However, face-to-face learning is effective only if teachers not only convey subject information to the students, but act as facilitators and guides of learning. In addition, student collaboration is useful when the more competent students helped the ones who faced difficulties in accomplishing their programming tasks.

### **Limitations of the Study**

The present study was a case study in which students in one course were studied. The sample size ( $n = 11$ ) may not be sufficient to adequately support a generalization of the evaluation results. Hence, successive cycles of experimentations, refinements, and evaluations of the blended learning model in future courses are warranted to generalize the findings of the present study. In addition, the methods used for collecting empirical data should be assessed and refined to ensure their quality, and eventually, completed with supplementary, both quantitative and qualitative, methods. The author intends to gather survey data in future courses in order to confirm the findings of this work, as well as to obtain a more in-depth theoretical understanding. It is also intended to determine key implications of the blended learning model for the practice of introductory programming in varied educational contexts, as well as to improve the quality of the model.

It is believed that the underlying Design-Based Research approach of this work has the potential to improving the overall quality of the blended learning model (The Design-Based Research Collective, 2003; Terashima, 2004). Through the iterative and continuous cycle of design, evaluation, and redesign in varied contexts, the author hopes to explore the design and evaluation processes of the learning model in more details and depth in order to gain theoretical and practical insights within computer programming, and, generate some evidence-based claims about the learning processes and further the current theoretical and practical knowledge of blended learning in higher education.

Finally, it is intended that issues that impact the lack of online discussion among students be explored in future research studies. Although the lack of online dialogue was not a key research question in the present study, the data collected indicate that online discussion did not play a key role in programming, since students did not view the lack of online discussion as a significant barrier to the learning process. However, the following questions remain: What should be done in face-to-face sessions and what should be delegated to online dialogue? Which motivational strategies are needed to engage students in online discussions? How to improve online collaboration and dialogue with the teacher and fellow students? These questions need to be addressed if the blended learning model has to achieve its promises of providing online dialogues that support effective learning.

## **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.
- Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45-73.
- Ben-David Kolikant, Y., & Pollack, S. (2004). Establishing computer science norms among high school students. *Computer Science Education*, 14(1), 21-35.

## Community of Inquiry

- Berglund, A., Daniels, M., & Pears, A. (2006). Qualitative research projects in computing education research: An overview. *Proceedings of the Eighth Australasian Computing Education Conference (ACE2006)*, Hobart, Tasmania, Australia, January 2006.
- Bonk, C. J., & Graham, C. R. (2006). *The handbook of blended learning: Global perspectives, local designs*. San Francisco, CA: Pfeiffer Publishing.
- Bruner, J. (1990). *Acts of meaning*. Cambridge, MA: Harvard University Press.
- Bryman, A. (2004). *Social research methods*. Second Edition. New York: Oxford University Press.
- Clancy, M., Titterton, N., Ryan, C.; & Slotta, J.; Linn, M.(2003). New roles for students, instructors, and computers in a lab-based introductory programming course. *Proceedings of SIGCSE'03*, February 19-23, Reno, Nevada, USA, 132-136.
- Conolly, T., & Standsfield, M. (2007). *Developing constructivist learning environments to enhance e-Learning*. In: N.A.Buzzetto-More, (Ed.) (2007). *Advanced principles of e-Learning* (pp. 19-38). Santa Rosa, California: Informing Science Press.
- Dagdilelis, V., Satratzemi, M., & Evangelidis, G. (2004). Introducing secondary education to algorithms and programming. *Education and Information Technologies 9* (2), 159-173.
- The Design-Based Research Collective (2003). Design-Based Research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1), 5-8.
- Dodero, J. M., Fernández, C., & Sanz, D. (2003). An experience on students' participation in blended vs. online styles of learning. *SIGCSE Bulletin 35*(4), 39-42.
- Duffy, T.M., Lowyck, J., & Jonassen, D.H. (1993). *Designing environments for constructive learning*. Berlin: Springer-Verlag.
- Dyson, M. C., & Campello, S. B. (2003). Evaluating virtual learning environments: What are we measuring? *Electronic Journal of E-learning 1*(1), 11-20.
- Exton, C. (2002). Constructivism and program comprehension strategies. *Proceedings of the 10<sup>th</sup> International Workshop on Program Comprehension (IWPC'02)*, La Sorbonne, Paris, France, June 26-29, 281-284.
- Gagne, E., Yekovich, C., & Yekovich, F. (1993). *The cognitive psychology of school learning* (2nd ed.). New York: HarperCollins.
- Gibbs, D. C. (2000). The effect of a constructivist learning environment for field-dependent/independent students on achievement in introductory computer programming. *SIGCSE Bulletin*, 03/00, Austin, Texas, USA, 207-211.
- Gonzales, G. (2004). *Constructivism in an introduction to programming course*. *JCSC 19*(4), April 2004, 299-305.
- Hadjerrouit, S. (1999). A constructivist approach to object-oriented design and programming. *Proceedings of ITCSE'99*, 6/99, Cracow, Poland, 171-174.
- Hadjerrouit, S. (2005). Web-based educational software in computer science: Technical and pedagogical usability. *Proceedings of ED-MEDIA 2005*, Montreal, Canada, June 27 – July 2, 1139-1144.
- Hadjerrouit, S. (2006). Introductory Java programming. Available at [http://fag.hia.no/kurs/inf100/www\\_docs](http://fag.hia.no/kurs/inf100/www_docs)
- Karagiorgi, Y., & Symeou, L. (2005). Translating constructivism into instructional design: Potential and limitations. *Educational Technology & Society*, 8(1), 17-27.
- Lin, B., & Hsieh, C. (2001). Web-based teaching and learner control: A research review. *Computers & Education*, 37(3-4), 377-386.
- Luca, J. (2006). Using blended learning to enhance teaching and learning. *Proceedings of the 8th Australian Conference on Computing Education*, 52, 3-4.

- Lui, A. K., Kwan, R., Poon, M., & Cheung, H. Y. (2004). Saving weak programming students: Applying constructivism in a first programming course. *SIGCSE Bulletin*, 36(2), 72-76.
- Mayes, J. T., & Fowler, C.J. (1999). Learning technology and usability: A framework for understanding courseware. *Interacting with Computers*, 11(5), 485-497.
- McDougali, A., & Boyle, M. (2004). Students' strategies for learning computer programming: Implications for pedagogy in informatics. *Education and Information Technologies* 9(2), 109-116.
- Mead, J. Gray, S., Hamer, J., James, R., Sorva, J., St.Clair, C. et al. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *Proceedings of ITiCSE '06*, June 26-28, Bologna, Italy, 182-194.
- Melis, E., & Weber, M. (2003). Lessons for (pedagogical) usability of e-Learning systems. *Proceedings of E-LEARN 2003*, Phoenix, Arizona, November 7-11, 281-284.
- Nilsen, J. (2000). *Designing web usability: The practice of simplicity*. New York: New Riders.
- Nokelainen, P. (2004). Conceptual definition of the technical and pedagogical usability criteria for digital learning material. *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21-26, 4249-4254.
- Pendergast, M. O. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education* 5, 491-515. Retrieved from <http://jite.org/documents/Vol5/v5p491-515Pendergast128.pdf>
- Piaget, J. (1969). *Judgment and reasoning in the child*. London: Routledge & Kegan Paul.
- Pollack, S., & Schertz, Z. (2003). Supporting project development in CS – the effect on intrinsic and extrinsic motivation. *Proceedings of the Eleventh International PEG Conference*, Russia.
- Roberts, G. (2003). Teaching using the web: Conceptions and approaches from a phenomenographic Perspective. *Instructional Science* 31, 127-150.
- Sajaniemi, J., & Kuittinen, M. (2005). An experiment on using roles of variables in teaching introductory programming. *Computer Science Education*, 15(1), 59-82.
- Schwieren, J., Vossen, G. & Westerkamp, P. (2006). Using software testing techniques for efficient handling of programming exercises in an e-Learning platform. *Electronic Journal of e-Learning (EJEL)*, 4(1), 87-94.
- Shaffer, S. C., & Lidwig, M.L. (2005). An online programming tutoring and assessment system. *SIGCSE Bulletin*, 37(2), 56-60.
- Shiratuddin, N., & Shahizan, H. (2003). A usability study for promoting eContent in higher education. *Educational Technology & Society*, 6(4), 112-114.
- Steffe, L.P., & Gale, J. (Eds.) (1995). *Constructivism in education*. New Jersey: Lawrence Erlbaum Associates.
- Storey, M.A., Phillips, B., Maczewski, M., & Wang, M. (2002). Evaluating the usability of web-based learning tools. *Educational Technology & Society*, 5(3), 91-100.
- Terashima, K. (2004). Blended learning for multimedia production course. *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21-26, 4049-4054.
- Vanhagen, S., & Zhou, G. (2004). Beyond usability: Evaluating instructional technology to better inform the field. *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21-26, 1120-1125.
- Vygotsy, L.S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.
- Wenger, E. (1998). *Communities of practice: Learning, meaning, and identity*. London: Pinter.
- Wang, F., & Hannafin, M. J. (2003). Importance of design-based research for technology-enhanced learning environments. *Proceedings of E-LEARN 2003*, Phoenix, Arizona, November 7-11, 1813-1816.

Wulf, T. (2005). Constructivist approaches for teaching computer programming. *SIGITE '05*, October 20-22, 2005, Newark, New Jersey, USA, 245-248.

## 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 and Sciences. He has been in the teaching profession for 25 years. He has extensive experience teaching object-oriented programming, Web design, database development, software engineering, and didactics of informatics. His research interests include object-oriented software development with the UML, computer science and software engineering education, didactics of informatics, use of ICT in mathematics education, development of e-Learning and Web-based learning systems.

Hadjerrouit has published over 45 papers in international journals and conference proceedings.