Informatics in Education, 2008, Vol. 7, No. 2, 181–210
© 2008 *Institute of Mathematics and Informatics, Vilnius*

Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study

Said HADJERROUIT
University of Agder, Faculty of Technology and Sciences
Serviceboks 422, N-4604 Kristiansand, Norway
e-mail: said.hadjerrouit@uia.no

Received: January 2008

Abstract. Blended 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 explore 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, implementation, and evaluation of the model and its implications for the learning of introductory computer programming.

Keywords: blended learning, computer programming, design-based research, e-learning, face-to-face learning, Java programming, learning cycle, online learning

1. Introduction

Teachers 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 difficulties of learning introductory programming for novice students remain to be researched. As blended learning becomes more and more pervasive in higher education as the most prominent delivery mechanism (Bonk and Graham, 2006), expectations for learning benefits in computer programming are becoming greater. But, just providing educators with a mix of face-to-face learning and information technologies, will not have the desired effect, if the underlying blended learning model does not rely on learning theory and pedagogical principles (Nocols, 2003). In addition, to gain practical insights, the model needs to be explored through successive cycles of implementation and evaluation in varied educational settings. A promising approach to explore blended learning is the design-based research paradigm.

The remainder of this article is organized as follows. First, the paper describes the design-based research approach and associated research questions of the work. This is followed by literature review. Third, the paper outlines a theoretical model of blended learning. In the next sections, the model is applied to an introductory programming course in Java. Finally, a summary of evaluation findings and implications for the design and delivery of blended learning in computer programming conclude the article.

2. 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 (Barab and Squire, 2004; The Design-Based Research Collective, 2003). 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. They are interdependent and reciprocal (Wang and 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 (Fig. 1):

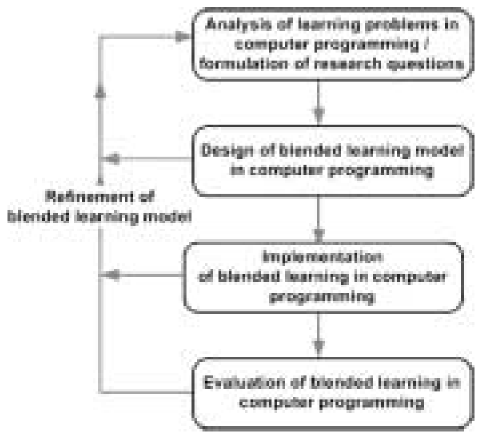
Analysis of learning problems in computer programming / formulation of research questions
Design of blended learning model in computer programming
Refinement of blended learning model
Implementation of blended learning in computer programming
Evaluation of blended learning in computer programming

Fig. 1. Design-based research as a feedback loop with four major phases.

1. Design-based research begins with the *analysis of the learning problems* associated with computer programming, the *formulation of research questions* of interest, and the review of relevant literature.
2. It continues with the *design of a blended learning model* that will be used to solve the learning problems. 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 and critical evaluation.

Analysis, design, implementation, and evaluation are interdependent and reciprocal. Refinements are continually made through successive cycles of experimentations, where the shortcomings of each cycle are identified, re-designed, re-implemented, and re-evaluated.

To explore the blended learning model in computer programming through successive cycles of experimentations, this work examined the following research questions:

- What are the benefits and barriers of learning computer programming for novice students in a blended learning environment?
- What are the critical factors of success in applying a blended learning model to computer programming and the implications for the design of blended learning?

3. Literature Review

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 and 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 *et al.*, 2004). According to Ben-Ari (2001), misconceptions are hard to change, unless students acquire a model of a computer.

For solving learning difficulties, educators apply learning theories to computer programming, in particular the constructivist learning theory (Ben-Ari, 2001; Exton, 2002; Gibbs, 2000; Gonzales, 2004; Hadjerrouit, 1999; Lui *et al.*, 2004; Mead *et al.*, 2006; Sajaniemi and 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, few educators systematically apply constructivism to the learning of computer programming (Berglund *et al.*, 2006). As a result, constructivist learning strategies to computer programming are only beginning to emerge.

Another solution to the learning of programming is to use information technologies that give appropriate feedback while working on programming assignments, e.g., online programming systems, Web-based programming tutors, online learning systems, or similar software. 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 on computer programming (Clancy *et al.*, 2003; Conolly and Stansfield, 2007; Hadjerrouit, 2005; Schwieren *et al.*, 2006; Shaffer and Lidwig, 2005). In addition, most applications focus on technological aspects rather than pedagogical principles.

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 and Graham, 2006). However, combining face-to-face learning with innovative information 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 (Dodero *et al.*, 2003). This is not sufficient to draw general conclusions about the effect of blended learning 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 problems and difficulties associated with the learning of introductory programming remain to be researched. Some educators believe that the solving of the problems associated with the learning of introductory programming requires a radical change from traditional instructional methods to constructivist learning environments and situated learning (Ben-Ari, 2004; Wulf, 2005), mostly because programming is considered as a skill that students need to acquire through an active construction process. Furthermore, according to Macdougali 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.

4. Blended Learning: Theoretical Model

According to (Anohina, 2005; Bonk and Graham, 2006; Nocols, 2003), there is no clear and unequivocal definition of the concept of blended learning. Definitions are partially exclusive and sometimes contradictory, and there are few common terms used consistently.

4.1. *Key Components of Blended Learning*

In the research literature it is difficult to distinguish the term “blended learning” from other terms such as “virtual 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 and 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 modalities, learning theories, and pedagogical dimensions
3. Combining e-learning with face-to-face learning

The third definition is the working definition of this work. It has a broader focus than the two first definitions 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 based on computers.

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

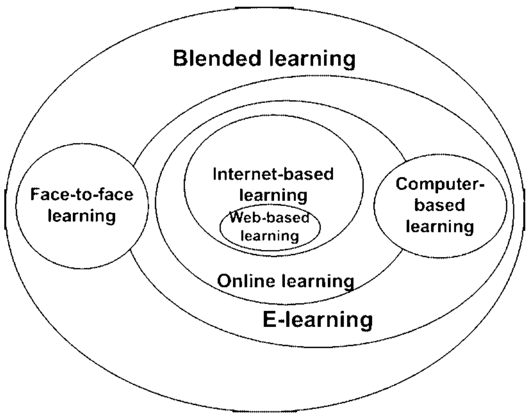

Fig. 2. Components of blended learning.

4.2. *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.

4.2.1. *Learning Theories*

The *cognitive learning theory* 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 learning theory* 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 *et al.*, 1993; Piaget, 1971; Steffe and 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.

The *socially situated learning theory* 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, 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.

4.2.2. *The Learning Cycle*

The literature on learning theories points to the fundamental philosophical differences between them (Lin and 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 (Karagiorgi and 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. The conceptualization phase is characterized by the process of interaction between the learners' pre-existing framework and teacher's knowledge. The construction phase 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. The dialogue phase is based on the socially situated learning theory as it is concerned with dialogue and collaboration.

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* consists of online dialogues between learners and teachers, as well as online group discussions and collaborations.

4.2.3. *Blended Learning and the Learning Cycle*

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. This results in blending at three different levels (Fig. 3):

- *Blending at the conceptualization phase*. Blending at this level occurs when the learning model 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.

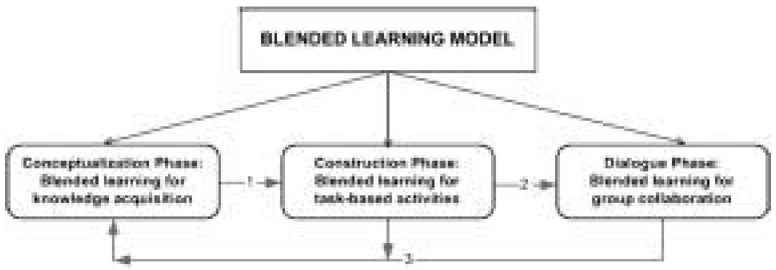
BLENDED LEARNING MODEL
Conceptualization Phase: Blended learning for knowledge acquisition
1
Construction Phase: Blended learning for task-based activities
2
Dialogue Phase: Blended learning for group collaboration
3

Fig. 3. Blended learning model with three iterative learning phases.

5. Applying the Blended Learning Model to Computer Programming

A case study approach was used to explore the application of the blended learning model to Java programming in 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. The course content was divided into 8 units (Table 1).

Table 1
Introductory Java programming: weekly schedule

Timeframe	Unit	Focus
Week 1	Unit 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	Unit 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	Unit 3	Control statements I: *if*-statements, *if-else*-statements, nested *if*-statements, *switch*-statements
Week 7		Control statements II: *while*-loop, *do-while*-loop
Week 8		Control structures III: *for*-loop, nested *for*-loop
Week 9	Unit 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	Unit 5	Strings: Text strings and basic operations, concatenation, and translations
Week 12	Unit 6	Applets: Sample Applets, executing Applets in the Applet-Viewer and in the Web browser, Applet Life Cycle
Week 13	Unit 7	Methods: Creating and calling methods, passing parameters, overloading methods, scope of local variables
Week 14	Unit 8	Repetition: Solutions of programming assignments and past exams
Week 15		Repetition: Solutions of programming assignments and past exams

5.1. *Conceptualization Phase: Programming 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 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 is a combination of *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. According to Hadjerrouit (1999), a number of pedagogical strategies can be used for teaching programming concepts:

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

5.2. *Construction Phase: Programming Activities*

The course was designed to support the construction phase of the learning model, that is to say the process of building computer programs through the performance of task-based activities. The process of constructing new programs is the product of 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 to the programming construction 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. According to Hadjerrouit (1999), the construction of computer programs requires the acquisition of higher-order thinking skills, such as:

a) *Analyze and design*. These skills 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 reuse skills.
c) *Study experts' programming solutions* from textbooks, and reuse them with slight modifications to solve similar problems.
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.

5.3. *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 programming 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, depending on the situation. According to Hadjerrouit (1999), a number of pedagogical strategies can be used for implementing this phase:

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*. Students usually 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.

5.4. *Online Resources*

Likewise, the online resources of the blended learning model were designed to promote the learning cycle with the three phases: conceptualization, construction, and dialogue. The platform of the online resources was a Web-based LMS (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. 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 strategies for designing the online resources for the conceptualization phase were:

- Break down programming knowledge into concepts that can be constructed using the pedagogical strategies described above.
- Provide a well-structured online presentation and organization of programming concepts.
- Provide user-friendly accessibility of the concepts and links to related study material.
- Provide a well-structured description of programming 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 programming concepts - in order to encourage students to construct computer 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 program 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 LMS-enabled discussions with the teacher and fellow students. Thus, the online resources were designed as *tertiary courseware* to support collaborative learning. This 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.

5.5. *The Blended Learning Model in Use*

The use of the blended learning model follows the learning cycle with three phases. First, 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 programming concepts. By using situated examples, the teacher enables the students to understand programming concepts.

Then, the students try to construct solutions to programming problems. For example, if the programming 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. Learning programming 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 programming 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.

6. Evaluation Methods

In this section, the data collection and analysis methods, the participants, the number of evaluation cycles, and the evaluation measures are described.

6.1. *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.

6.2. *Data Collection and Analysis Methods*

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

- A pre-questionnaire was provided to the students after one month of 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 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 2004.

6.3. *Evaluation Cycles*

According to the design-based research paradigm, it may be necessary to continually refine the blended learning model through successive cycles of evaluations, where the shortcomings of each cycle are identified, re-designed, re-implemented, and re-evaluated. The number of cycles depends on the course duration, which was three months in this case study. As a result, two evaluation cycles were performed during this short timeframe: A pre- and a post-evaluation. Hence, the evaluation findings resulted from:

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

6.4. *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 questionnaire by placing a cross "X" in the appropriate box using the scale provided.

7. Pre-evaluation and Redesign of the Blended Learning Model

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

a) The process of interaction between students' pre-existing knowledge and the level of difficulty, scope, and depth of the concepts presented in 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:

- A number of students found that the level of difficulty, scope, and depth of the subject matter compared to the students' background knowledge was 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 felt that Java programming was 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 contained useful resources for 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.

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 programming concepts and the acquisition of Java 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 should be 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 both in the classroom and computer lab, where students performed their programming activities.

8. 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. It addressed both technical and pedagogical issues:

- Technical evaluation issues addressed the extent to which the online resources helped the students to learn computer programming, and were useful as learning resource.
- Pedagogical evaluation issues addressed the extent to which the blended learning model provided support to the learning process.

According to the research literature (Melis and Weber, 2003; Nokelainen, 2004), it is important to link technical issues to pedagogical considerations when evaluating blended learning, because the primary goal of the blended learning model is to minimize 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 blended learning model. Hence, the starting point for specifying the pedagogical evaluation 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.

8.1. *Students' Perceptions of the Online Resources of Blended Learning*

The evaluation of the online resources is a widely used method to assess online learning (Agostinho and Herrington, 2004; Nilsen, 2000; Shiratuddin, Hassan & Landoni, 2003; Storey *et al.*, 2002). The following evaluation addressed the extent to which the online resources provided technical support to Java programming. The evaluation was concerned with 7 variables. The statistical analysis is shown in Table 2.

The evaluation of the online resources shows the following results. As Table 2 indicates, most students were globally satisfied with the online resources. All students strongly agreed or agreed that they were satisfied with variable 1, 2, 3, 4, and 6 of the online resources (content, navigation, usability, structure, global satisfaction). Moreover, 10 students recommended the reuse of online resources in future courses (variable 5). In addition, 63.63% of the students disagreed or strongly disagreed that the online resources should be improved (variable 7), indicating that the quality of the resources was good. From the results it can be inferred that the online resources were well implemented.

In addition to these issues, the students were asked to answer two questions about the usefulness of online resources for learning Java. Responses are presented below in Table 3 (1= "Very Useful"; 2 = "Useful"; 3 = "Not Very Useful"; 4 = "Didn't Use Them").

Table 2

Evaluation of online resources

Variables	**N**	**Min**	**Max**	**Mean**	**SD**
1. The content is concise and well-organized	11	4	5	4,55	0,522
2. The navigation is straight and intuitive	11	4	5	4,45	0,522
3. The online resources are well-organized, user-friendly, and easy to access	11	4	5	4,55	0,522
4. The online resources are well-structured in a clear and understandable manner	11	4	5	4,27	0,467
5. I recommend the reuse of the online resources in future courses	11	3	5	4,36	0,674
6. I am globally satisfied with the online resources	11	4	5	4,36	0,505
7. The online resources are fine. They don't need to be improved	11	1	5	3,64	1,120

Table 3

Usefulness of online resources

Variables	**N**	**Min**	**Max**	**Mean**	**SD**
8. How much the online resources helped you to learn Java?	11	3	4	3,45	0,522
9. How useful were the online resources as a learning resource?	11	2	4	3,36	0,674

In response to the first question (variable 8), all students believed that the online resources were very useful (45.45%) or useful (54.54%) for learning Java programming. In response to the second question (variable 9), most students felt that the online resources were very useful (45.45%) or useful (45.45%) as a learning resource. Only one student didn't believe that the resources were useful.

Finally, the students were asked about the uses of the online resources (Table 4). From the results it appears that 54.54% of the students indicated that they 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

Table 4

Uses of the online resources

10. How did you use the online resources?	**Frequency**
1. I used all the relevant learning resources each week	6
2. I 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

(18.18%) for the exam (Mean=4.09, Min=3, Max= 5, SD=0.701).

From the results, it can be inferred that the online resources were very useful or useful for the learning of Java concepts and programming, and that most students used the relevant resources each week or the ones needed to help with programming tasks.

8.2. *Students' Perceptions of Pedagogical Issues of Blended Learning*

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.

8.2.1. *Conceptualization Phase*

This phase included 9 evaluation variables. The results are displayed in Table 5.

Analysis of the responses to the categories "Strongly Agree" (code 5) and "Agree" (code 4) 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" (code 2), one can be satisfied with the evaluation results. These findings are clearly reflected in the statistical analysis. Hence, it can be inferred from the students' responses that the conceptualization phase was globally well implemented as it provided appropriate support to the understanding of Java programming concepts and performance of programming tasks.

Table 5

Evaluation of the conceptualization phase

Variables	**N**	**Min**	**Max**	**Mean**	**SD**
11. Overall, I am satisfied with the course content.	11	3	5	4,09	0,539
12. The course helped me to construct my own knowledge and understanding of Java programming.	11	4	5	4,36	0,505
13. The course provided appropriate support to help me complete programming tasks.	11	3	5	3,91	0,701
14. The course supported my motivation and engagement in learning Java programming.	11	3	5	3,64	0,674
15. The course provided motivating programming tasks.	11	2	5	3,64	0,809
16. The knowledge demands and the level of difficulty of the course are appropriate.	11	3	5	3,64	0,674
17. The course content is closely aligned with the intended course objectives and goals.	11	3	5	3,64	0,809
18. Programming tasks enabled me to reflect on and consolidate my learning of programming at various stages throughout the semester.	11	4	5	4,36	0,505
19. The course textbook helped me to learn programming and to solve programming tasks.	11	1	4	3,45	0,934

Table 6

Evaluation of the construction phase

Variables	N	Min	Max	Mean	SD
20. I analyze programming tasks before programming them.	11	1	4	2,82	0,982
21. I design an algorithm for programming tasks before programming them.	11	1	4	2,27	0,786
22. I make connections to previous programming solutions when performing programming tasks.	11	2	5	3,73	0,905
23. I make connections to my own previous programming solutions when doing programming tasks.	11	4	5	4,45	0,522
24. I reuse programming knowledge and solutions when performing programming tasks.	11	2	5	3,73	1,009
25. I develop alternative (multiple) solutions to programming tasks.	11	1	4	2,45	0,934
26. I compare and contrast the solutions to find out the most efficient one.	11	1	5	2,73	1,348
27. I make predictions about the program's behaviour before testing the program solution.	11	2	5	3,18	0,874
28. I feel I have a sense of control over my own learning when demonstrating the programming solution to the teacher.	11	3	5	3,82	0,603
29. I test the programming solution with a set of data.	11	3	5	4,09	0,539
30. I reflect on and evaluate the programming solution process.	11	3	4	3,36	0,505

8.2.2. *Construction Phase*

The construction phase included 11 variables. The evaluation was concerned with the students' programming construction process and the acquisition of higher-order skills, such as problem analysis, design and reuse skills, making connections between concepts, explaining, justifying, making predictions beforehand, developing alternative solutions, reflecting on and evaluating the solution process. The evaluation results are displayed in Table 6.

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, reusing programming knowledge and solutions, demonstrating the solution process, and program testing (Variable 22, 23, 24, 28, and 29). Otherwise the evaluation indicates lower mean to the categories "Strongly Agree" or "Agree" with regard to the skills: analyzing the problem, designing algorithms, developing alternative solutions, comparing and contrasting, making predictions, and reflecting and evaluating, (Variable 20, 21, 25, 26, 27, and 30).

8.2.3. *Dialogue Phase*

The dialogue phase included 7 evaluation variables. It was concerned with teacher-student interactions and student-student collaborations. The statistical analysis is shown in Table 7.

Summarizing, in some contrast to the construction phase, the dialogue phase was

Table 7

Evaluation of the dialogue phase

Variables	**N**	**Min**	**Max**	**Mean**	**SD**
31. The teacher is always well-prepared for giving lectures and supervising programming tasks.	11	3	5	4,09	0,701
32. Teacher's lecturing (PP slides, teaching Java on the blackboard, dialogue with students) helped me to understand the key concepts of Java	11	3	5	4,27	0,647
33. Discussions and dialogue with the teacher helped me identify the strengths and limits of my programming knowledge.	11	2	5	3,91	1,044
34. There were a variety of opportunities to interact with the teacher and to ask questions.	11	2	5	3,91	0,831
35. The teacher helped me to solve the programming tasks.	11	3	5	4,18	0,874
36. Student demonstrations of programming tasks and teacher's explanations are helpful for understanding Java programming.	11	3	5	4,18	0,751
37. I work together and collaborate with other students when solving programming tasks.	11	3	5	4,45	0,688

quite good implemented since most students perceived that teacher-student interactions were beneficial to them in terms of discussions, collaborations, teacher's preparation and lecturing, students' demonstrations and teacher's explanations. Students felt that they were engaged in their learning of programming in collaboration with the teacher. They also benefited from their collaborations with fellow students. These findings are clearly reflected in the statistical analysis.

8.3. *Correlation Analysis*

To complete the analysis of the data collected it is important to connect the online resources to the pedagogy of the blended learning model. This is the result of correlation analysis between the 10 variables of the online resources (7+3 additional variables regarding the usefulness and uses of the online resources) and those of the conceptualisation (9 variables), construction (11 variables), and dialogue phases (7 variables).

8.3.1. *Positive Correlations*

The study revealed that the variables of the online resources positively correlated with 7–8 (out of 9) variables of the conceptualization phase and 5–7 (out of 7) variables of the dialogue phase of the blended learning model. In the case of the construction phase, there was a positive correlation with 4–7 (out of 11) variables, otherwise the correlation was negative. The results of correlation analysis are given in Table 8. The numbers without parentheses represent the number of positive correlations and those in parentheses represent the total number of variables.

In the case of the construction phase, the variables of the online resources positively correlated with the following variables: connections to previous knowledge, connection

Table 8

Correlation analysis: Number of positively correlated variables in relation to the total number of variables

Variables	Conceptualization	Construction	Dialogue
1. The content is concise and well-organized	7 (9)	5 (11)	6(7)
2. The navigation is straight and intuitive	7(9)	5(11)	5(7)
3. The online resources are well-organized, user-friendly, and easy to access	8(9)	6(11)	7(7)
4. The online resources are well-structured in a clear and understandable manner	8(9)	6(11)	7(7)
5. I recommend the reuse of the online resources in future courses	8(9)	6(11)	7(7)
6. I am globally satisfied with the online resources	8(9)	6(11)	6(7)
7. The online resources are fine. They don't need to be improved	7(9)	7(11)	5(7)
8. How much the online resources helped you to learn Java?	8(9)	7(11)	6(7)
9. How useful were the online resources as a learning resource?	7(9)	6(11)	6(7)
10. How did you use the online resources?	7(9)	4(11)	6(7)

to own previous knowledge, reusing previous knowledge and solutions, having a sense of control, and in less degree testing the programming solution. Otherwise, the correlation was basically negative for other variables of the construction phase.

Additionally, the results of correlation analysis revealed that there was a strong positive correlation between the online resources and a number of pedagogical variables. The strongest positive correlations are described as follows.

The variable "*The content of the online resources is concise and well-organized*" was positively correlated with the following variables:

- The course helped me to construct my own knowledge and understanding of Java programming ($r = 0.690$, $p < 0.05$).
- I reuse programming knowledge and solutions when performing programming tasks ($r = 0.690$, $p < 0.05$).
- Discussions and dialogue with the teacher helped me identify the strengths and limits of my programming knowledge ($r = 0.650$, $p < 0.05$).

The variable "*The navigation is straight and intuitive*" was positively correlated with the following variables:

- The course helped me to construct my own knowledge and understanding of Java programming ($r = 0.828$, $p < 0.01$).
- I make connections to previous programming solutions when performing programming tasks ($r = 0.712$, $p < 0.05$).
- Discussions and dialogue with the teacher helped me identify the strengths and limits of my programming ($r = 0.633$, $p < 0.05$).

The variable "*The online resources are well-organized, user-friendly, and easy to ac-*

cess” was positively correlated with the following variables:

- I feel I have a sense of control over my own learning when demonstrating the programming solution process to the teacher ($r = 0.664$, $p < 0.05$).
- The teacher is always well-prepared for giving lectures and supervising programming tasks ($r = 0.671$, $p < 0.05$).
- Discussions and dialogue with the teacher helped me identify the strengths and limits of my programming ($r = 0.650$, $p < 0.05$).
- Student demonstrations of programming tasks and teacher’s explanations are helpful for understanding Java programming ($r = 0.742$, $p < 0.05$).

The variable “*The online resources are well-structured in a clear and understandable manner*” was positively correlated with the following variables:

- The knowledge demands and the level of difficulty of the course are appropriate ($r = 0.664$, $p < 0.01$).

The variable “*I recommend the reuse of the online resources in future courses*” was positively correlated with the following variables:

- I feel I have a sense of control over my own learning when demonstrating the programming solution process to the teacher ($r = 0.671$, $p < 0.05$).
- Discussions and dialogue with the teacher helped me identify the strengths and limits of my programming ($r = 0.620$, $p < 0.05$).
- Student demonstrations of programming tasks and teacher’s explanations are helpful for understanding Java programming ($r = 0.647$, $p < 0.05$).

The variable “*I am globally satisfied with the online resources*” was positively correlated with the following variables:

- The course provided appropriate support to help me complete programming tasks ($r = 0.669$, $p < 0.05$).

The variable “*The online resources are fine. They don’t need to be improved*” was positively correlated with the following variables:

- The course helped me to construct my own knowledge and understanding of Java programming ($r = 0.669$, $p < 0.05$).
- The course content is closely aligned with the intended course objectives and goals ($r = 0.612$, $p < 0.05$).
- I test the programming solution with a set of data ($r = 0.722$, $p < 0.05$).

Strong positive correlations mean that the variation in the variables of the online resources and some pedagogical variables are very closely connected, but that there is some influence of other variables in the extent to which they vary (Bryman, 2004). A possible explanation of these strong positive correlations was that the online resources had a positive impact on the students’ learning of Java programming for the following reasons:

- The level of difficulty of the subject matter was minimised by well-organized and easy accessible online information.
- The understanding of Java programming concepts was supported by well-structured online information.

- Discussions and dialogues with the teacher took into consideration the availability of online information that could be accessed at any time for revision.
- Demonstrating the programming solution process and teacher's explanations relied on online information in terms of well-designed examples.
- Teacher's lecturing in classroom was supported by online resources that could be accessed at any time and any place.
- Connections to previous programming knowledge and reuse of Java program code were facilitated by information available online.
- Program testing was supported by online programming examples and Java program code that could be executed directly.

8.3.2. *Negative Correlations*

Correlation analysis shows that the relationship between the variables of the online resources and those of the construction phase was either negative or virtually zero, but rarely positive. This was the case of 6 variables: problem analysis, design of algorithms before coding, develop alternative solutions, compare and contrast solutions, making predictions beforehand, and reflect on the solution process (Table 9). If the correlation is virtually zero at approximately (0.050, 0.060, −0.017, or −0.039), then the variation of each variable is associated with other variables than the one presented (online resources) in this analysis.

A possible explanation of the negative correlation is that there is a tendency such that, the more a student is able to acquire higher-order thinking skills, the less likely he or she needs support from the online resources. Put another way, the more a student needs support from the online resources, the less likely he or she is able to acquire higher-order thinking skills. Another possible explanation is that the online resources have insignificant or no impact (if the correlation is virtually zero) on the acquisition of higher-order thinking (or fundamental) skills, such as problem analysis, design algorithm, make predictions, develop alternative solutions, and reflect on the solution process. In this case, the acquisition of fundamental programming skills requires cognitive efforts rather than the support of well-designed online resources.

8.4. *Supplementary Evaluation: Exam Scores, Discussions, and Observations*

To strengthen the validity of the evaluation results the author considered supplementary evaluation: exam scores, informal discussions with the 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 exam. 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

Table 9

Negative correlation between the online resources and some variables of the construction phase

	Analyze Problem	**Design Algorithm**	**Develop Alt. Solutions**	**Compare/ Contrast**	**Make Prediction**	**Reflect on Solution**
1. The content is concise and well-organized	$r = -0,567$	$r = -0,155$	$r = -0,149$	$r = -0,478$	$r = -0,458$	$r = -0,069$
2. The navigation is straight and intuitive	$r = -0,408$	$r = -0,089$	$r = -0,261$	$r = -0,374$	$r = -0,418$	$r = -0,311$
3. The resources are well-organized, user-friendly, and easy to access	$r = -0,177$	$r = -0,155$	$r = -0,312$	$r = -0,620*$	$r = -0,458$	$r = 0,311$
4. The resources are well-structured in a clear & understandable manner	$r = 0,555$	$r = 0,050$	$r = -0,313$	$r = -0,188$	$r = -0,379$	$r = -0,039$
5. I recommend the reuse of the online resources in future courses	$r = -0,343$	$r = -0,017$	$r = -0,447$	$r = -0,650*$	$r = -0,293$	$r = 0,160$
6. I am globally satisfied with the online resources	$r = -0,055$	$r = 0,229$	$r = 0,039$	$r = -0,134$	$r = -0,165$	$r = -0,179$
7. The online resources are fine. They don't need to be improved	$r = -0,521$	$r = 0,351$	$r = -0,312$	$r = 0,060$	$r = 0,483$	$r = 0,257$
8. How much the online resources helped you to learn Java?	$r = -0,408$	$r = 0,399$	$r = -0,056$	$r = -0,658$	$r = 0,239$	$r = 0,449$
9. How useful were the online resources as a learning resource?	$r = -0,041$	$r = 0,171$	$r = -0,289$	$r = -0,540$	$r = 0,216$	$r = 0,160$
10. How did you use the online resources?	$r = 0,317$	$r = 0,132$	$r = -0,528$	$r = -0,500$	$r = -0,030$	$r = -0,103$

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, students 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 in classroom and computer lab 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 appeared that the students' programming activities were computer-dependent, even if many students were unable to deal with the compilation errors. Nevertheless, most 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.

9. Discussion

In this section, a summary of findings of the case study and the associated implications for the learning of computer programming in blended learning environments are presented. The limitations of the study and recommendations for future research are discussed as well. The findings help to answer the two research questions:

- What are the benefits and barriers of learning computer programming for novice students in a blended learning environment?
- What are the critical factors of success in applying a blended learning model to computer programming and the implications for the design of blended learning?

9.1. *Summary of Findings*

The benefits and barriers of learning computer programming for novice students in the blended learning environment are summarized as follows.

First, the students displayed a high level of satisfaction with the online resources of the blended learning model. Most students strongly agreed or agreed that the online resources were globally well-designed with regard to content, navigation, usability, and structure. Moreover, most students recommended the reuse of online resources in future courses, indicating that the quality of the resources was good. This is clearly an improvement compared to the programming course given in 2004 (Hadjerrouit, 2005).

Second, the results of the study revealed that the conceptualization phase was globally well implemented as it provided appropriate support to the understanding of Java concepts and performance of programming tasks. Students agreed that the conceptualization phase helped them to construct their own knowledge of programming and provided motivating tasks. Students pointed out that the course content was closely aligned with the intended course objectives and goals and that the knowledge demands and the level of difficulty of the course were appropriate.

Third, the findings indicated that the students were satisfied in their efforts to acquire skills associated with connecting to previous knowledge and programming solutions, reusing programming knowledge and solutions, demonstrating the solution process, and program testing. Otherwise the findings indicated lower results for the skills associated with analyzing the problem, designing algorithms before coding, comparing and contrasting solutions, reflecting on the solution process, making predictions beforehand, and developing alternative solutions.

Fourth, the students pointed out that the dialogue phase was well implemented, because most students perceived that teacher-student interactions were beneficial for them.

Students felt that they were engaged in their learning of programming in interaction with the teacher. They also benefited from their collaborations with fellow students. These findings are clearly reflected in the statistical analysis of the study. Nevertheless, 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 than online discussions.

Finally, the results of correlation analysis revealed that the online resources were positively correlated with most variables of the conceptualization and dialogue phases. This indicated that there was a clear connection between the variables of the online resources and those of the conceptualization and dialogue phases. Likewise, there was a clear relationship between the resources and some variables of the construction phase (connecting to previous knowledge and solutions, reusing knowledge, have sense of control, program testing), otherwise the correlation was basically negative or virtually zero. This was the case of fundamental programming activities, such as analysis, design, making predictions, comparing and contrasting, and developing alternative solutions.

9.2. *Critical Factors of Success and Implications for the Design of Blended Learning*

From the evaluation of the blended learning model in computer programming the following critical factors of success and implications for the design of blended learning can be drawn.

The findings of this study indicated 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 indicated that the supporting and facilitating students' online learning might be of great value for acquiring some programming skills. The implication of this finding is that well-designed online resources are a critical factor of success in applying the blended learning model. But, online resources in themselves are not sufficient, unless they are designed according to well-known software usability criteria and pedagogical principles. Well-designed online resources can provide support at three different levels:

- First, when online resources are well-designed as a source for subject information, they can provide useful support to 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 programming concepts.
- Second, 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 can provide support through the adaptation, modification, and reuse of well-designed programming examples and their solutions, as well as reusable program code available online. Clearly, well-designed online resources can help students to acquire reuse skills and connect new knowledge to previous knowledge.

- Third, interactions by means of email or discussion forum are still important, but 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.

The second implication of the study is that collaborative activities are more important to novice students entering the field of computer programming than the individual acquisition of higher-order thinking skills. This confirms the evidence found in the research literature that programming is an inherently social activity as good programs are developed not in isolation; instead they involve interaction with other people (Macdougali and Boyle, 2004). Collaborative activities might be of great value for the learning process when two conditions are fulfilled. First, the teacher provides appropriate help through face-to-face discussion and supervision. Second, the students receive a greater amount of guidance in programming activities. The implication of this finding is that blended learning at any level should promote collaborative work, giving the students a sense of how programming activities can be performed in interaction with fellow students and teachers.

The third implication of this study is that the acquisition of fundamental programming skills requires cognitive efforts rather than the support of well-designed online resources. Fundamental skills are those related to problem analysis, algorithm design, comparing and contrasting solutions, developing multiple and alternative solutions, making predictions, and evaluating and reflecting on the solution process. As a result, this study suggests that more learning theories and pedagogical practices be explored on how to promote the acquisition of fundamental skills.

Finally, the fourth implication that follows from the findings of this study is that a pedagogically sound model of blended learning has the potential for improving the learning of computer programming. But teachers need to be aware of the limitations of blended learning. Online resources are highly important but not sufficient to help students progress beyond the novice stage to higher-order skills. This because, contrary to online resources as a source for subject information, which can be designed according general software usability criteria and pedagogical principles, 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 follow students. To alleviate this difficulty, online learning needs to be combined with a pedagogically sound model of face-to-face-learning, which includes both student-student collaborations and teacher-students interactions. 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-student collaboration is particularly useful when the more competent students help the ones who face difficulties in accomplishing their programming tasks.

9.3. *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 findings and associated implications for blended learning. It is possible that the use of other online resources and within another group of students there are other results and implications. Three factors are warranted to generalize the findings of the present study. First, successive cycles of experimentations and evaluations of the blended learning model in future courses. Second, the methods used for collecting survey data should be assessed to ensure their quality and completed with supplementary, both quantitative and qualitative, methods. Finally, the application of the blended learning model for the practice of introductory programming in varied educational contexts.

10. Conclusion and Future Work

Through the iterative and continuous cycle of design, evaluation, and redesign in varied contexts (Design-Based Research Collective, 2003), the author hopes to explore the blended learning model in more details and depth in order to further the current theoretical and practical knowledge of blended learning in higher education. It is also intended that issues that impact the lack of online dialogue be explored in future case studies. Although the lack of online dialogue was not a key research question in the present study, the data collected indicate that online dialogue did not play a key role in programming, since students did not view the lack of online dialogue as a significant barrier to the learning process. However, the following questions need to be addressed if the blended learning model has to achieve its promises of providing online dialogues: What should be done face-to-face and what should be delegated to online dialogue? Which motivational strategies are needed to engage students in online dialogue? How to improve online collaboration and dialogue with the teacher and follow students? Finally, it is important to explore the extent to which online learning can provide support to the acquisition of higher-order thinking skills in computer programming.

References

Anohina, A. (2005). Analysis of the terminology used in the field of virtual learning. *Educational Technology and Society*, **8**(3), 91–102.

Agostinho, S., and J. Herington (2004). An effectiveness evaluation of online learning environment. In *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21–26, pp. 3476–3481.

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

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, **20**(1), 45–73.

Ben-Ari, M. (2004). Situated learning in computer science education. *Computer Science Education*, **14**(2), 85–100.

Ben-David Kolikant, Y., and S. Pollack (2004). Establishing computer science norms among high school students. *Computer Science Education*, **14**(1), 21–35.

Berglund, A., M. Daniels and A. Pears (2006). Qualitative research projects in computing education research: an overview. In *Proceedings of the Eighth Australasian Computing Education Conference* (*ACE2006*), Hobart, Tasmania, Australia, January 2006.

Bonk, C.J., and C.R. Graham (2006). *The Handbook of Blended Learning: Global Perspectives, Local Designs*. Pfeiffer Publishing, San Francisco, CA, USA.

Bruner, J. (1990). *Acts of Meanin*g. Harvard University Press, Cambridge, MA.

Bryman, A. (2004). *Social Research Methods*. Oxford University Press, Oxford.

Clancy, M., N. Titterton, C. Ryan, J. Slotta and M. Linn (2003). New roles for students, instructors, and computers in a lab-based introductory programming course. In *Proceedings of SIGCSE'03*, February 19–23, Reno, Nevada, USA, pp. 132–136.

Conolly, T., and M. Standsfield (2007). Developing constructivist learning environments to enhance e-learning. In N.A. Buzzetto-More (Ed.), *Advanced Principles of e-Learning*. Informing Science Press, Santa Rosa, California, USA, pp. 19–38.

Dagdilelis, V., M. Satratzemi and G. Evangelidis (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., C. Fernández and D. Sanz (2003). An experience on students' participation in blended vs. online styles of learning. *SIGCSE Bulletin*, **35**(4), 39–42.

Duffy, T.M., J. Lowyck and D.H. Jonassen (1993). *Designing Environments for Constructive Learning.* Springer-Verlag, Berlin.

Dyson, M.C., and S.B. Campello (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. In *Proceedings of the 10th International Workshop on Program Comprehension* (*IWPC'02*), La Sorbonne, Paris, France, June 26–29, pp. 281–284.

Gagne, E., C. Yekovich and F. Yekovisch (1993). *The Cognitive Psychology of School Learning* (2nd ed.). HarperCollins, New York.

Gibbs, D.C. (2000). The effect of a constructivist learning environment for field-dependent/independent students on achievement in introductory computer programming. In *SIGCSE Bulletin*, 03/00. Austin, Texas, USA, pp. 207–211.

Gonzales, G. (2004). Constructivism in an introduction to programming course. *JCSC*, **19**(4), 299–305.

Hadjerrouit, S. (1999). A constructivist approach to object-oriented design and programming. In *Proceedings of ITCSE'99*, 6/99, Cracow, Poland, pp. 171–174.

Hadjerrouit, S. (2005). Web-based educational software in computer science: technical and pedagogical usability. In *Proceedings of ED-MEDIA 2005*, Montreal, Canada, June 27–July 2, pp. 1139–1144.

Hadjerrouit, S. (2006). *Introductory Java Programming*. `http://fag.hia.no/kurs/inf100/www_docs`

Karagiorgi, Y., and L. Symeou (2005). Translating constructivism into instructional design: potential and limitations. *Educational Technology and Society*, **8**(1), 17–27.

Lin, B., and C. Hsieh (2001). Web-based teaching and learner control: a research review. *Computers and Education*, **37**(3–4), 377–386.

Luca, J. (2006). Using blended learning to enhance teaching and learning. In *Proceedings of the 8th Australian Conference on Computing Education*, vol. 52, pp. 3–4.

Lui, A.K. *et al.* (2004). Saving weak programming students: applying constructivism in a first programming course. *SIGCSE Bulletin*, **36**(2), 72–76.

Mayes, J.T., and C.J. Fowler (1999). Learning technology and usability: a framework for understanding courseware. *Interacting with Computers*, **11**(5), 485–497.

Macdougali, A., and M. Boyle (2004). Students strategies for learning computer programming: implications for pedagogy in informatics. *Education and Information Technologies*, **9**(2), 109–116.

Mead, J. *et al.* (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. In *Proceedings of ITiCSE'06*, June 26–28, Bologna, Italy, pp. 182–194.

Melis, E., and M. Weber (2003). Lessons for (pedagogical) usability of e-learning systems. In *Proceedings of E-LEARN 2003*, Phoenix, Arizona, November 7–11, pp. 281–284.

Nilsen, J. (2000). *Designing Web Usability*: *The Practice of Simplicity.* New Riders, New York.

Nokelainen, P. (2004). Conceptual definition of the technical and pedagogical usability criteria for digital learning material. In *Proceedings of ED-MEDIA 2004*, Lugano, Switzerland, June 21–26, pp. 4249–4254.

Nocols, M. (2003). A theory of e-learning. *Educational Technology and Society*, **6**(2), 1–10.

Pendergast, M.O. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education*, **5**, 491–515.

Piaget, J. (1969). *Judgment and Reasoning in the Child.* Routledge & Kegan Paul, London.

Pollack, S., and Z. Schertz (2003). Supporting project development in CS – the effect on intrinsic and extrinsic motivation. In *Proceedings of the Eleventh International PEG Conference*, St Petersburg, Russia.

Roberts, G. (2003). Teaching using the web: conceptions and approaches from a phenomenographic perspective. *Instructional Science*, **31**, 127–150.

Sajaniemi, J., and M. Kuittinen (2005). An experiment on using roles of variables in teaching introductory programming. *Computer Science Education*, **15**(1), 59–82.

Schwieren, J., G. Vossen and P. Westerkamp (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., and M.L. Lidwig (2005). An online programming toturing and assessment system. *SIGCSE Bulletin*, *37*(2), 56–60.

Shiratuddin, N., and H. Shahizan (2003). A usability study for promoting eContent in higher education. *Educational Technology & Society*, **6**(4), 112–124.

Steffe, L.P., and J. Gale (Eds.) (1995). *Constructivism in Education*. Lawrence Erlbaum Associates, New Jersey.

Storey, M.A., B. Phillips, M. Maczewski and M. Wang (2002). Evaluating the usability of web-based learning tools. *Educational Technology & Society*, **5**(3), 91–100.

Vygotsy, L.S. (1978). *Mind in Society*: *The Development of Higher Psychological Processes.* Harvard University Press, Cambridge, MA.

Wenger, E. (1998). *Communities of Practice*: *Learning, Meaning, and Identity*. Pinter, London.

Wang, F., and M.J. Hannafin (2003). Importance of design-based research for technology-enhanced learning environments. In *Proceedings of E-LEARN 2003*, Phoenix, Arizona, November 7–11, pp. 1813–1816.

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

S. 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 University of Agder, 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 26 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 60 papers in international journals and conference proceedings.

Kompiuterių programavimo mokymo ir mokymosi įvairialypis modelis: atvejo tyrimas

Said HADJERROUIT

Aukštąjį mokslą vis labiau vilioja įvairialypio mokymosi (anglų k. blended learning) modelis, kadangi vis labiau prieinamos tampas inovatyvios informacinės technologijos. Tačiau tik paprastai derinant akivaizdinį mokymą su nuotolinėmis studijomis panaudojant informacines technologijas neįmanoma pasiekti efektyvaus mokymo rezultatų ir efektyvių mokymosi sprendimų. Kad suderintas mokymasis būtų sėkmingas, jis turi remtis mokymosi teorija ir kruopščiai parengta pedagogine strategija. Norint tyrinėti įvairialypio mokymosi modelį naudojant bandymų ciklą, reikia įvertinti jo trūkumus. Tam būtinas priėjimas prie mokslinių tyrimų bazės. Šiame straipsnyje pateikiami tyrimo duomenys, atlikti taikant įvairialypį mokymosi modelį mokant Java programavimo pagrindų. Aptariams modelio sudarymas, realizacija ir įvertinimas, taip pat jo taikymas mokantis kompiuterinio programavimo pradmenų.