Creative Education
2012. Vol.3, Special Issue, 773-783
Published Online October 2012 in SciRes (http://www.SciRP.org/journal/ce)
http://dx.doi.org/10.4236/ce.2012.326116

Student-Centered Learning Objects to Support the Self-Regulated Learning of Computer Science

Ali Alharbi, Frans Henskens, Michael Hannaford
School of Electrical Engineering and Computer Science, The University of Newcastle, Newcastle, Australia
Email: Ali.H.Alharbi@uon.edu.au, Frans.Henskens@newcastle.edu.au, Michael.Hannaford@newcastle.edu.au

Received August 30th, 2012; revised September 30th, 2012; accepted October 14th, 2012

The most current computing curriculum guidelines focus on designing learning materials to prepare students for lifelong learning. Under the lifelong learning paradigm, students are responsible for controlling and monitoring their learning processes. This undoubtedly includes the ability to choose suitable learning materials. Correspondingly, instructional paradigms are shifting from teacher-centered to more student-centered models that require students to be self-regulated learners. On the other hand, recent trends in learning materials' instructional design focus on moving toward the concept of Learning Object-based instructional technology. A learning object is a unit of instruction with a specific pedagogical objective that can be used and reused in different learning contexts. Designing learning objects to support students in their self-regulated learning is not an easy task due to the lack of underlying pedagogical frameworks. It is difficult to find learning objects related to students' specific preferences and requirements. In this study, a number of learning objects are designed to support the self-regulated learning of programming languages concepts based on the theory of learning styles. Students' interactions with these learning objects are managed using an online learning object repository. The repository helps students identify their preferred learning styles and find the relevant learning objects. The results of the evaluations of these learning objects revealed that students perceive them to be easy to use and effective in supporting their learning about different programming languages concepts.

Keywords: Learning Objects; Learning Styles; Self-Regulated Learning; Computer Science Education

Introduction and Problem Statement

The current trend in instructional design is moving toward the concept of the "learning object". Learning objects are learning materials with pedagogical objectives that are intended for use and reuse in different learning contexts (Sosteric & Hesemeier, 2002). The number of learning resources stored in online learning object repositories has increased dramatically. Many learning object repositories that store a large number of learning objects for use in different disciplines are available today. Learning objects are designed to be used by students and instructors in their learning and teaching. Therefore, pedagogy should be the primary factor taken into consideration in the design and delivery of learning objects to learners. Learning theories are at the core of any pedagogical framework, and learning objects represent a new era in instructional technology that aims to support teaching and learning in many disciplines.

On the other hand, in recent years, the concept of self-regulated learning has received increasing attention in educational research, especially in higher education. Self-regulated learning provides students with the ability to control and monitor their learning processes and determine how to locate the suitable learning resources (Pintrich & Zusho, 2007). Self-regulated learning is an active topic in education research due to its importance to academic success and lifelong learning (Dettori & Persico, 2008). Correspondingly, instructional paradigms are shifting to more student-centered instead of teacher-centered models (Berglund et al., 2009). This implies that students must become more self-regulated learners. The recent computing curricula guidelines produced by the IEEE/ACM stress the importance of designing computer science learning materials that help prepare computer science students for lifelong learning (Sahami, Guzdial, McGettrick, & Roach). To this end, the learning materials should take the diversity of students' preferences and requirements into consideration. However, there is a scarcity of instructional design theories that guide the design and use of learning objects (Wiley et al., 2004).

The primary goal of learning object instructional technology is to simplify and enhance the process of the instructional design and distribution of learning materials (Wiley et al., 2004). In its simplest form, instructional design theory is about helping people learn better; it involves the processes of analyzing learners' requirements and designing learning materials to satisfy these requirements (Reigeluth, 1999). The instructional design of learning objects differs significantly from the conventional instructional design process. Shifting toward self-regulated learning paradigms stresses the importance of designing learning objects that are compatible with students' needs and preferences and improving students' interactions with these objects.

Theoretical Background

Learning Object Instructional Technology

Different definitions of the term "learning object" can be found in the literature. The IEEE Learning Technology Standards Committee (LTSC) define the learning object as "any entity, digital or non-digital, which can be used, re-used or

referenced during technology supported learning" (LTSC, 2002). This definition is broad, suggesting that anything used for education could be considered a learning object. However, many of the subsequent definitions of the term "learning object" are attempts to narrow the scope of the IEEE LTSC definition. Wiley excluded non-digital items from the IEEE definition, describing a learning object as "any digital resource that can be reused to support learning" (Wiley, 2000). Sosteric and Hesemeier (Sosteric & Hesemeier, 2002) synthesized the various definitions of learning object, defining a learning object as any item with a pedagogical objective that is intended for used and reuse in different learning contexts.

A learning object is a collection of learning items such as images, animations, simulations and other resources to form a complete learning unit. In terms of instructional technology, the learning object is described as the "technology of choice in the next generation of instructional design, development, and delivery, due to its potential for reusability, adaptability, and scalability" (Wiley, 2002). Learning objects are grounded in the object-oriented paradigm of computer science. In this context, a learning object can be viewed as an encapsulated unit of instruction that can be used independently or in tandem with other learning objects to achieve a specific pedagogical goal. According to the object-oriented paradigm, objects are created based on templates known as classes. A class is an abstract representation of a set of objects. Objects of the same class are abstractly the same; they share the same general characteristics and differ only in the values of their attributes and their behaviors.

Learning Theories: Educational Paradigm Shift

Behaviorism is a theory in educational psychology that emphasizes observable events (Watson, 1997). This theory focuses on the learner's behavior and excludes the analysis of consciousness, which it views as a concept unsuitable for scientific research. According to this theory, learning is considered to be a change in the learner's behavior that occurs as a result of external events. Behaviorists do not go so far as to deny the existence of the consciousness, but they claim that the inner activities of the mind are essential aspects of behavior that can only be studied and described in terms of their external indicators. In Watson's view, "psychology should restrict itself to examining the relation between observable stimuli and observable behavioral responses" (Thagard, 2010). According to behaviorism, the teacher is dominant in the learning process, and the learner has a more limited role.

Behaviorism was the dominant psychological theory until it was replaced by cognitive theory in what is referred to as the "cognitive revolution". Cognitive psychologists "have proposed that the mind contains such mental representations as logical propositions, rules, concepts, images, and analogies, and that it uses mental procedures such as deduction, search, matching, rotating, and retrieval" (Thagard, 2010). They view psychology as the science of cognition: that is, the study of "thinking and the mental processes humans use to solve problems, make decisions, understand new information or experiences, and learn new things" (Weinstein & Acee, 2008).

Bruning et al. (1999) have identified six important themes of cognitive psychology theories that are of particular relevance to educators: 1) Knowledge is constructed based on the interaction between the learners' current knowledge and the new information they encounter; 2) Cognitive theories emphasize the concept of schema, or the representational frameworks we use to translate sensory impressions into our personal interpretations of reality; 3) Cognitive theories advance the idea that learners are reflective and self-directed in their learning. In short, cognitive theories consider metacognition to be a key component of education; learners use strategies to control the learning process; 4) Cognitive theories associate learners' motivations and beliefs with learning outcomes; 5) Cognitive theory stresses the importance of social interaction to cognitive growth. By interacting with their peers and instructors, learners may gain perspectives that can either provide them with new learning experiences or shape their learning approaches and strategies; 6) Cognitive theories that view the mind as similar to a computer are tempered by their acceptance of the concept of contextualism, the position that one's perceptions of external situations come into play in the processes of comprehension and memory encoding.

Learning Styles

Learning style theory is among the learning theories that have arisen from the cognitive revolution. Learning styles describe some of the individual differences that may influence the development of self-regulated learning strategies; learning style can be defined as "a particular way in which an individual learns" (Pritchard, 2009). One of the most comprehensive definitions of learning style was provided by Keefe (Keefe, 1988), who defined it as "the characteristic cognitive, affective and psychological behaviors that serve as relativity stable indicators of how learners perceive, interact with and respond to the learning environment". Adopting a specific teaching or instruction style without being aware of students' different learning styles may lead to inefficient learning outcomes for some students (Pritchard, 2009). Teachers should be aware of their students' learning styles and vary their teaching strategies and materials to be compatible with different styles. However, many teachers attempt to differentiate their teaching materials according to difficulty levels, not according to their students' learning styles.

Felder-Silverman Learning Style Model

The Felder-Silverman Learning Style Model (Felder & Silverman, 1988) is a learning style model used to identify learning styles, especially in science and engineering education. The Index of Learning Styles (ILS) is the instrument used to identify learning styles based on this model. The model consists of four dimensions (Felder & Silverman, 1988) (see **Figure 1**).

Perception: This dimension describes the type of information an individual perceives preferentially. Sensing learners prefer concrete content and facts and are detail oriented, whereas intuitive learners prefer abstract concepts, theories and mathematical formulas and dislike details. The sensing learner tends to solve problems using well-established methods and dislikes complications. The intuitive learner appreciates innovation and new methods of solving problems and dislikes repetition.

Input: This dimension describes the type of presentation an individual prefers. Visual learners prefer learning through visual media, such as pictures, charts and diagrams, whereas verbal learners prefer spoken or written materials and explanations. Both types of learners learn better when the material is deliv-

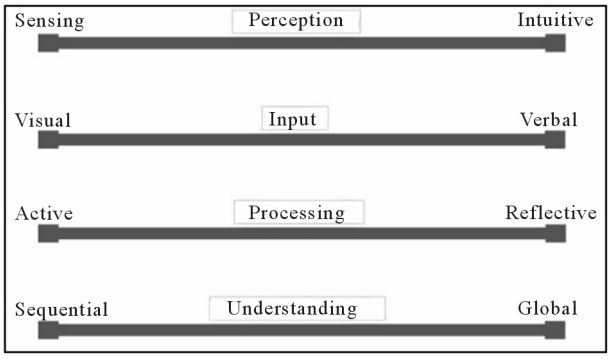

Figure 1.
Felder-Silverman learning style model.

ered using visual, verbal and written forms. Learners of both types can help themselves by finding relevant explanations of the subjects discussed in class in their textbooks or external resources, which may be suggested by their teachers.

Processing: This dimension describes how the learner processes information. Active learners prefer learning in groups, and they tend to try new things, whereas reflective learners prefer working alone and tend to think about how things work before attempting them.

Understanding: This dimension describes how the learner progresses toward understanding information. Sequential learners prefer following a logical, step-by-step linear approach, whereas global learners prefer to absorb the learning materials randomly, in large jumps, without following a step-by-step approach, until they grasp the full picture. Global learners can fix a complex problem once they grasp the full picture, but they might encounter difficulties when attempting to describe how they solved it. Courses are normally taught according to a sequential presentation format. Sequential learners can learn effectively under this method of instruction if they attempt to connect the learning materials logically and develop outlines for the lectures by consulting their teachers or references. Global learners need to grasp the full picture before going into the details; therefore, it may be helpful for them to skim through the content of each chapter or unit of study to gain an overview and try to link the new content to something they already know.

Research in Learning Styles

Education researchers agree that there are different learning styles that must be accommodated to improve the teaching and learning process. In addition, empirical studies that have been conducted to investigate the implications of different learning styles on students' performance have found that there are significant differences in the levels of academic achievement of students with different learning styles (Akdemir & Koszalka, 2008; Alharbi, Paul, Henskens, & Hannaford, 2011; Mills, Ayre, Hands, & Carden, 2010; Zander et al., 2009). One explanation for this result is that the learning materials favor specific learning styles and ignore others.

There appears to be a debate on how to integrate learning styles into curriculum design and teaching and learning activities. The lack of empirical studies that evaluate the effectiveness of learning styles-based interventions in the educational process in many subjects has made it difficult to generate recommendations for teachers and curriculum designers. The research on learning styles focuses primarily on the identification of students' learning styles and how this might affect their academic achievements. In addition, the research on learning styles follows a track that is isolated from other educational theories. The role of learning styles in self-regulated learning has not been investigated and appears to offer a potential direction for future research.

The main hypothesis that dominates the research on learning styles is called the "matching hypothesis" (Coffield, 2004). This hypothesis argues that if a learner is presented with learning material that is compatible with his/her own learning style, his/her learning process improves. Further, teaching methods that are mismatched with the learner's style might lead to difficulties in learning. However, research on how this could be applied in context to improve the teaching and learning process in many disciplines, including computer science, is scarce. "Learning style awareness" was put forward in response to critical reviews of learning style theories as an alternative and promising hypothesis for future research on learning styles (Coffield, 2004; Coffield, Moseley, Hall, & Ecclestone, 2004). This hypothesis claims that knowledge of learning styles should be used to increase self-awareness, which leads to improvements in the learning and teaching process. Learners who become aware of their learning styles are more likely to be aware of their strengths and weaknesses and, therefore, will have more control of their learning processes. In addition, teachers who are aware of the diversity of learning styles among their students are most likely to adopt different teaching approaches that appeal to different types of students. This study adopts a framework to achieve balance between these two learning style hypotheses.

Related Work

This section reviews the literature on the difficulties related to the teaching and learning of computer science concepts and the diversity of students' learning styles.

Difficulties in Teaching and Learning Computer Science Concepts

The research on computer science education has investigated various issues related to the difficulties involved in the teaching and learning of computer science concepts. This includes investigating the misconceptions in students' understandings of certain difficult computer science concepts. This review focuses on teaching and learning the concepts of programming languages and paradigms.

Teaching object-oriented concepts appears to be a difficult task for the teachers who must find the best way to teach them and the students who are asked to understand many concepts concurrently (Milne & Rowe, 2002).

Many empirical studies have found that students encounter difficulties and develop misconceptions related to the understanding of object-oriented concepts such as encapsulation, inheritance and polymorphism (Fleury, 2001; Holland, Griffiths, & Woodman, 1997; Ragonis & Ben-Ari, 2005a, 2005b). The concepts of inheritance and polymorphism are essential to the understanding of the object-oriented paradigm. In a recent study, Liberman, Beeri, and Ben-David Kolikant (Liberman, Beeri, & Ben-David Kolikant, 2011) noted that despite the fact

that inheritance and polymorphism are central to object-oriented paradigm, research has only recently begun to address students' difficulties with and misconceptions of these concepts. They classified students' difficulties understating inheritance and polymorphism into four distinct clusters based on the reasons behind their difficulties: alternative programming models, analogies, misunderstandings of inheritance and misunderstandings of basic object-oriented concepts.

Or-Bach and Lavy (Or-Bach & Lavy, 2004) conducted a study to investigate students' misconceptions of inheritance and abstraction. The study participants were thirty-three college students who had completed two courses in object-oriented programming and design. The students were given the task of designing an inheritance hierarchy to demonstrate their understanding of fundamental object-oriented concepts, such as inheritance and polymorphism. Based on the analysis of the students' solutions, the study results show that the students did not attain the intended level of abstraction. The study proposed a taxonomy of the task analysis regarding abstraction and inheritance. The taxonomy consists of three levels, each representing a degree of abstraction that the students should reach when analyzing the task. The study offers some recommendations for instructional design. Examples of model answers should be presented to students; in addition, the students should be given the chance to discuss their solutions for different problems and encouraged to participate in self-assessment and reflection on the various solutions.

In (Benaya & Zur, 2008) and (Hadar & Leron, 2008), the researchers found that novices are not the only students who encounter difficulties understanding object-oriented concepts. Both students in advanced courses and experts appear to have similar difficulties. Benaya and Zur (Benaya & Zur, 2008) present the results of a study conducted to identify students' misconceptions of object-oriented concepts in an advanced programming course. The study participants were 39 students. The course's final exam was used as a research instrument, and it consisted of a number of questions related to object-oriented programming. The study revealed various misconceptions related to object-oriented programming. These misconceptions were related to misunderstandings of some aspects of inheritance and polymorphism, such as the chain of constructor calls in the object creation process and dynamic binding. Based on these results, the study suggested improving the course's instructional method by incorporating more concrete examples and using visualization tools to help students understand the dynamic processes during the execution of the program. Hadar and Leron (Hadar & Leron, 2008) investigated the difficulties faced by experts participating in object-oriented concept workshops. The authors used cognitive psychology to uncover the causes of these difficulties. The studies revealed some of the problems participants faced during the study, which were related to identifying objects and confusion between concrete and abstract classes. Using dual-processing theory, the authors claimed that these difficulties were a result of the clash between the formal object-oriented concepts and their intuitive origins. "Under the force of these general cognitive mechanisms, deciding on appropriate objects, classes, and relations is sometimes influenced by irrelevant surface clues or everyday meanings of these concepts, thus leading to inappropriate choices."

In addition to understanding object-oriented concepts, it is essential for computer science students to be familiar with other concepts related to programming languages. These concepts include memory management and parameter passing methods.

Research in computer science education has been criticized for its lack of reference to established pedagogical theories (Holmboe, McIver, & George, 2001). Fjuk, Bennedsen, Berge, and Caspersen argue that past research and course designs in computer science education have not explicitly described theoretical foundations related to learning theories; the field appears to focus on the technology rather than educational theory (Fjuk, Bennedsen, Berge, & Caspersen, 2004).

As a result of this criticism, the computer science education research community has begun to focus on investigating contemporary educational theories and their potential roles in improving the teaching and learning methods used in computer science. Over the last few years, learning theories related to student-centered approaches to learning have received attention in computer science education. These approaches focus on the learner's role in discovering and constructing knowledge through active participation in the learning process. Student-centered educational paradigms place a high level of responsibility on learners to self-regulate their learning using different learning strategies. Learners should plan for their learning by better utilizing the available learning resources and monitoring their progress toward achieving their goals. Under this new educational paradigm, it is essential for teachers to both master the subject matter and understand the different ways students come to understand different concepts, planning their teaching methods to take these differences into account (Holmboe et al., 2001). According to some computer science education researchers (Ben-Ari, Berglund, Booth, & Holmboe, 2004), it is essential to understand how students learn about computer science concepts and the conditions of learning. To achieve this goal, they suggest studying theories that describe the mental models students create to describe a target system; this mental model does not necessarily reflect the actual system. The same authors suggest using variations in presenting the problem under investigation to allow students to experience the problem from different perspectives.

Learning Styles in Computer Science Education

This section reviews the research related to learning styles in computer science education. Such a review provides insight into whether the diversity in learning styles of computer science students is being taken into consideration in instructional methods. Previous studies have investigated computer science students' learning styles. Thomas et al. (Thomas, Ratcliffe, Woodbury, & Jarman, 2002) investigated the learning styles of students enrolled in an introductory programming course. The majority of students in the study were assessed to be sensing, visual, reflective and sequential. The result of the study indicated that in the exam portion of the course, significant differences were detected in the students' performance between the reflective and active learners in favor of the reflective learners and between verbal and visual learners in favor of the verbal learners. One interesting result was that although the majority of students were visual learners, the verbal learners exhibited the highest performance in the course. This result is consistent with the results reported in (Chamillard & Karolick, 1999) and (Allert, 2004). In a recent study (De Raadt & Simon, 2011), the authors stated that research on the exploration of learning styles in computer science education is scarce. This motivated them to conduct a study investigating students' learning styles in an

introductory programming course. The study found that the majority of students preferred practical applications and concrete information connected to reality and were comfortable with details. They preferred to learn using simulation and case studies.

Pedagogical Framework to Support Computer Science Learning Objects

Critical systematic reviews of learning style theories stress the importance of future research on learning styles focusing on increasing students' awareness of their preferred learning styles. This study proposes a pedagogical framework to improve the design of computer science learning objects to reflect the shift toward student-centered educational paradigms. Under the proposed framework, learning objects are designed to support different aspects of students' learning styles. However, the students are not restricted to specific learning objects; instead, they are responsible for self-regulating their learning with the help of a collaborative learning object repository that helps increase their awareness of their learning styles. The proposed framework consists of a number of dimensions, each focusing on specific aspects of these learning styles (**Table 1**).

Level of Abstraction Dimension

This dimension focuses on reducing abstraction in computer science learning objects by introducing more features that link the abstract concepts to the real world. Many computer science concepts are illustrated using program codes. The level of abstraction of these concepts can be reduced by showing the dynamic changes in the code using animation.

Presentation Dimension

This dimension focuses on varying the presentation of learning objects' content to address the diversity of students' learning styles. This dimension balances between the visual and verbal presentations of the content of the learning object.

Level of Interactivity Dimension

This dimension focuses on the interactive features that the learning object offers to students. There should be a balance between interactive features that allow students to discover the ideas behind the concepts themselves and the students' ability to think about how the concept works: for example, stop-and-think questions.

Sequencing and Organization Dimension

This dimension is associated with the structure of learning object content to achieve a balance between treating the content in sequential order and enabling the students to grasp the big picture as early as possible.

The Framework in Action: A Case Study in Computer Science Education

Programming Languages and Paradigms

The course covers different topics that are essential for any computer science and software engineering students to study. A course that addresses the theories behind the design and implementation of programming languages is an integral part of any computer science and software engineering program (IEEE/ACM, 2005). Programming language concepts are presented by comparing the features of different programming languages, such as Java and C++. In addition, different programming paradigms are discussed and compared.

Collaborative Learning Object Repository

To support students' self-regulated learning, a collaborative learning object repository has been developed. All the learning objects have been stored in this repository and made available for students' use. The repository implements a module responsible for identifying students' learning styles and providing them with a learning guide to help them increase their awareness of their learning styles and develop more control over their learning processes.

A number of learning objects have been developed to support the course on programming languages and paradigms. All these objects have been published in the collaborative learning object repository to allow students to use them to support their self-regulated learning. The designs of the learning objects are

Table 1.
Learning style-based pedagogical framework for computer science learning objects.

Dimension	Design criteria	Examples related to computer science education
Level of abstraction	• Examples or analogies to connect the abstract concept to the real world. • Mathematical formula and/or program code	The concepts of inheritance and polymorphism can be described using real-world examples, such as animals and vehicles. The program code can be added later.
Presentation	• Pictures and diagrams • Animations and simulations • Textual and audio descriptions	Linked-list operations can be animated by highlighting the code step by step and showing the dynamic changes in the list. Each step is described using text and audio.
Interactivity	• Interactive animations • User control • Self-assessment with instant feedback and model answers	The animation used in the linked list has controls through which the user can select which operation to apply and pause, resume, or rewind the animation at any time.
Sequencing and organization	• Clear, sequential order when covering the concepts • Overview (big picture) and summary • Comparisons	The learning object that describes polymorphism starts with an overview of polymorphism and the content of the learning object. There is an option to show a comparison between Java and C++ in the implementation of polymorphism. At the end, there is a summary of the contents of the learning object.

based on templates to ensure that the dimensions proposed in the framework are covered. These learning objects cover different topics in the course.

Memory Management

It is essential for programs to allocate memory to store data values and structures. If a program allocates a memory and never releases it, it may cause problems with insufficient memory. Programming languages provide different types of support for memory management. Understanding memory management is essential to computer science and software engineering students in their early programming careers. It is important for any programmer to learn about the different approaches adopted by programming languages to manage the allocation and release of memory.

The memory can be divided into three main regions: Static, Stack and Heap. The static region is initialized at load time, its lifetime is the complete program run and its visibility is the whole program. Heap is a memory region that is allocated and de-allocated under the program's control during its run time. The heap memory is allocated when it is needed; it is the programming environment's responsibility to keep track of what areas of memory are free and what is currently in use to release memory when it is no longer needed. Stack is the memory region used to support procedure calls by storing local variables and parameters. Stack memory is deleted automatically when it is no longer needed.

Programming languages adopt different approaches to memory allocation. In Java, all objects are allocated to the heap and released automatically by the garbage collection when they are no longer needed. Only primitives and references can reside in the stack; allocating objects to the stack is not supported in Java. In contrast, memory management in the C++ programming language is different. In C++, it is the programmer's responsibility to apply an efficient memory management practice by keeping track of the memory that is no longer needed and releasing it.

In this section, a number of learning objects are designed to help students understand concepts related to memory management by comparing the memory management approaches taken by the Java and C++ programming languages (e.g., **Figure 2**).

The learning object uses animation to present the dynamic allocation of the stack and heap memory during program execution. The animation provides a comparison between Java and C++ with regards to memory management.

To illustrate the concept of information hiding, a learning object has been developed based on a real-world example. This learning object is an animation of Java code used to compare different access modifiers to help students grasp and retain the concept of information hiding in the object-oriented programming paradigm.

Data Structures in Java

Data structure is an essential subject for all computer science and software engineering students. Linked lists, stacks and queues are the main data structure types covered in this course. These data structures are implemented differently in Java than in C++ and other languages. A number of learning objects have been developed to describe how different operations on data structures work dynamically, using step-by-step animations and highlighting in the code (e.g., **Figure 3**).

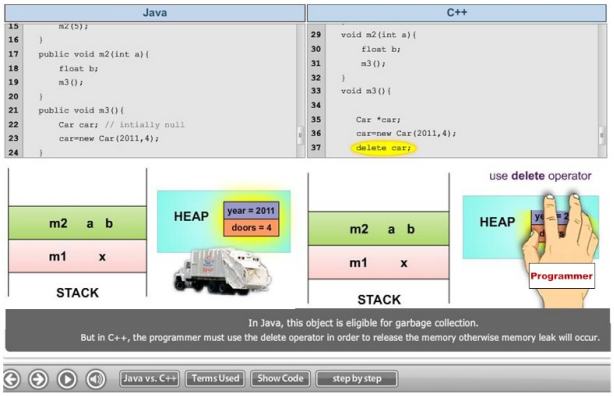

Figure 2.
A learning object to teach memory management concepts.

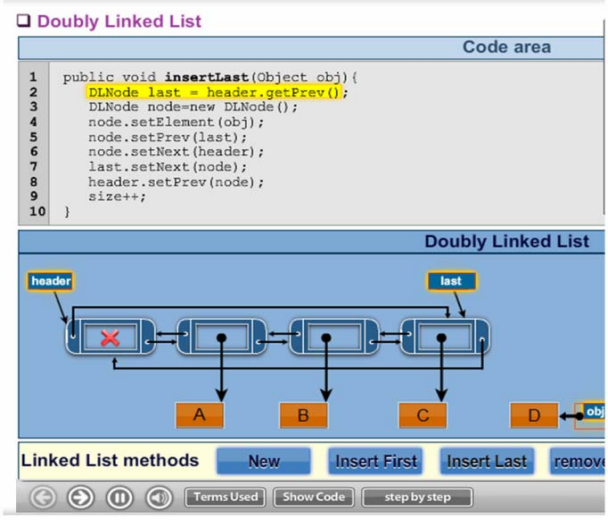

Figure 3.
Doubly linked list learning object.

Object-Oriented Concepts

Many empirical studies have found that students have difficulties with and misconceptions in their understanding of object-oriented concepts such as encapsulation, inheritance and polymorphism (Fleury, 2001; Holland et al., 1997; Ragonis & Ben-Ari, 2005a, 2005b). Encapsulation and information hiding are two essential object-oriented concepts that refer to the bundling of data with the methods operating on those data. External objects can only interact with an encapsulated entity through its interface. A proper understanding of encapsulation is important for novice programmers to gain essential object-oriented programming skills. Fleury (Fleury, 2001) conducted a study to gain insight into the different ways students understand encapsulation. The study found that a lack of experience in object-oriented programming leads novice students to focus on tracing the code, rather than on long-term benefits related to code reuse and maintenance.

To illustrate the concepts of encapsulation and information hiding, a learning object based on a real-world example has been developed. The learning object is an animation of Java code that presents a comparison between different access modifiers to help students grasp and retain the concept of informa-

tion hiding in the object-oriented programming paradigm.

The concepts of inheritance and polymorphism are essential to an understanding of the object-oriented paradigm. Inheritance refers the ability to create a new piece of code based on an existing one instead of creating the code from scratch. This increases the maintainability of the software. Polymorphism is an object-oriented concept referring to an object's ability to respond to a message according to its own interpretation of that message. The same message can elicit different responses from different objects. This makes it easy to extend the program code by adding new object types without the need for major modifications of the code.

A number of learning objects have been developed to support students' understanding of inheritance and polymorphism. Each learning object starts with a real-world example to make it easy for students to grasp the concepts as early in the process as possible.

Parameter Passing Methods

Programming languages use different methods to pass parameters into procedures. A learning object has been designed to help students to understand the dynamic process of different parameter passing methods in an animated step-by-step manner (**Figure 4**). The learning object depicts the code of the caller and the called procedures and highlights the code statement that is being executed. Students can choose among the different parameter passing methods.

Concurrency

Concurrency is one of the important characteristics of any modern operating system today. It allows multiple processes to be executed simultaneously. In concurrent processing systems, issues of process cooperation and competition arise. Mutual exclusion and synchronization are the main concurrency concepts discussed in any introductory operating systems course. When more than one process needs to share the same resource, mutual exclusion must be guaranteed: that is, only one process at a time can use the shared resources.

Synchronization refers to the coordination of activities between multiple processes through the sharing of information.

Programming languages provide different techniques to achieve synchronization between processes. The programming languages and paradigm course shows how the synchronization between processes can be managed using different techniques, such as semaphores, monitors and message passing. These solutions are normally applied to different classical synchronization problems, such as producer-consumer, readers-writers and dining-philosophers. Enforcing mutual exclusion between processes can introduce critical problems, such as deadlock.

A number of learning objects have been designed to describe the concepts related to concurrency in Java. These include a learning object that describes the semaphore mechanism to achieve synchronization between processes and prevent problems with concurrency. **Figure 5** shows a screen shot of a learning object used to describe the bounded-buffer problem.

Participants

The participants in this study are 36 students enrolled in the *Programming Languages and Paradigms* course at the University Of Newcastle, Australia, in the first semester of 2012. This course is a second-year course that is required for computer science and software engineering students. It covers different programming language concepts, such as memory management, inheritance, polymorphism and parameter passing methods, and some programming paradigms, such as concurrent programming.

Data Collection Instruments

The data collection instruments in this study include the Index of Learning Styles (ILS) and the Self-Regulated Learning Strategies questionnaire. To evaluate the educational effectiveness of the learning objects, students' ratings of and comments on the learning objects in the repository have been collected. In addition, the students completed an online questionnaire at the end of the semester designed to gather information about their perceptions of the usefulness of the learning objects and the online repository used in the study.

Index of Learning Styles

The ILS instrument is used to identify the learning styles of students based on the Felder-Silverman Learning Style Model (Felder & Soloman, 1997). The instrument consists of 44 items that identify the students' learning styles based on four dimensions: 1) Sensitive-Intuitive; 2) Visual-Verbal; 3) Active-Reflective; and 4) Sequential-Global.

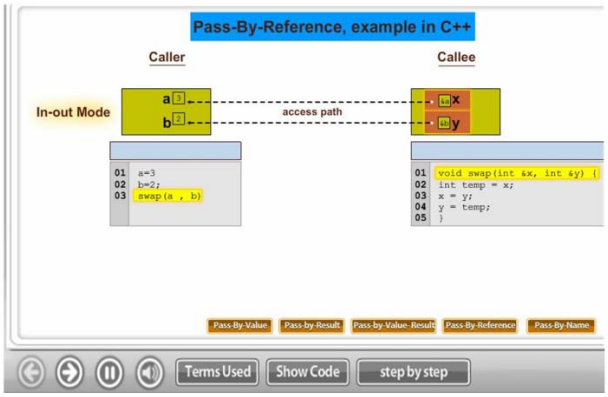

Figure 4.
A learning object to describe parameter passing methods.

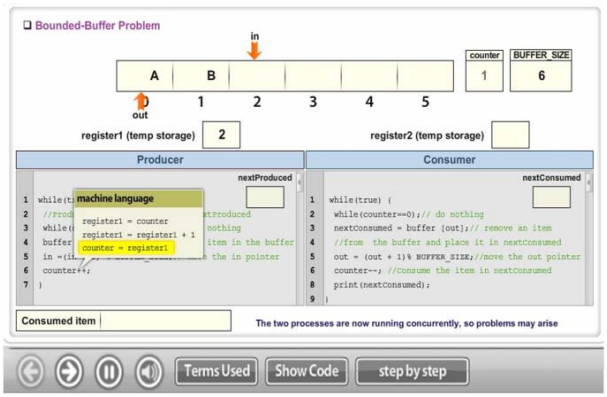

Figure 5.
A learning object used to describe the bounded-buffer problem.

Learning Object Ratings and Open-Ended Comments

The repository provides each student with the ability to rate learning objects according to a 5-point scale to indicate their educational effectiveness. In addition to their ratings, the students can provide open-ended qualitative comments on their perceptions and satisfaction level after using the learning objects.

Student Satisfaction Questionnaire

This instrument is an online questionnaire completed by the students as part of the overall feedback questionnaire to evaluate the educational effectiveness of the entire learning object repository at the end of the semester. The questionnaire measures the students' satisfaction in terms of their perceptions of the collaborative learning object repository. The questionnaire uses a 7-point Likert scale in which 1 represents *strongly disagree* and 7 represents *strongly agree.* The questionnaire measures students' satisfaction with different aspects of the learning objects.

Method and Procedure

The students were issued accounts to use the repository for self-regulated learning. The system was used for approximately 10 weeks, until the end of the semester. Students completed the learning style and self-regulated learning questionnaire at the beginning of the semester and all the questionnaire responses were entered into the repository. However, if a student logged into the system for the first time, he or she would be redirected to complete the learning style and self-regulated learning questionnaires online if he or she had not completed the paper-based questionnaire.

Results

This section presents the results of the analysis of the data collected in this study. First, the distribution of students' learning styles is presented, followed by the self-regulated learning strategies reported by the participants. Second, the analysis of the data collected from the students' usage of the learning object repository is presented.

Students' Learning Styles

Table 2 presents the students' learning styles based on the four dimensions of the Felder-Silverman Learning Style Model. In the perception dimension, the majority of students (66.7%) were sensing learners, while 33.3% were intuitive learners. In the input dimension, 83.3% of the students were categorized as visual learners, while only 16.7% were verbal learners. The processing dimensio categorizes students based on how they process information. The majority of the students were reflective learners (63.3 %), while 36.1% were categorized as active learners. Finally, in the last dimension, the students were categorized based on how they progressed towards an understanding of the learning material. The majority of students were sequential learners (61.1%), while 38.9% were global learners.

Educational Effectiveness of Learning Objects

A total of 34 students used the system during the semester. Students viewed the learning objects inside the system and interacted with them to learn about different topics. This section presents the results of the educational effectiveness evaluations of the learning objects.

Table 2.
Students' distributions based on their preferred learning styles.

Dimension	Learning Style	No. of Students (n)	Percentage (%)
Perception	Sensing	24	66.7
	Intuitive	12	33.3
Input	Visual	30	83.3
	Verbal	6	16.7
Processing	Active	13	36.1
	Reflective	23	63.9
Understanding	Sequential	22	61.1
	Global	14	38.9

Learning Object Ratings

Table 3 presents the average ratings of the learning objects in the *Programming Languages and Paradigms* course. **Figure 6** depicts the average ratings of the learning objects grouped by topic. As the table demonstrates, the overall average rating for all the learning objects was 4.18 out of 5. The learning objects related to the topic of arrays in Java and C++ had the highest average rating (5.0), followed by the singly linked list (4.9). The learning objects related to the diamond inheritance problem had the lowest average rating (3.5).

Students' Perceptions of the Learning Objects

Table 4 presents the students' responses to the questions related to students' perceptions of the learning objects. As the table shows, the mean responses range from 3.89 to 6.37 (out of 7). The overall mean is 5.76, indicating that students have positive attitudes toward using these learning objects; they perceive them as useful to support their self-regulated learning. The students perceive the highlighting of the code during the animation as the most useful feature for them (mean = 6.37). This is followed by the dynamic descriptions that appear during each step of the animation (mean = 6.16) and the real-world examples used in the learning objects to describe the abstract concepts (mean = 6.05).

Qualitative Comments on Learning Objects

In addition to the learning object ratings and the perception questionnaire, a qualitative evaluation was conducted to gather some feedback on the usefulness of these learning objects. The students were asked to provide comments on the educational effectiveness of the learning objects. The focus of the evaluation was to obtain insight into the students' perceptions of and satisfaction with the learning objects, their willingness to use them in their studies, and the features that make the students want to use the learning objects. The instrument also asked the students to identify any obstacles that might limit the learning objects' benefits and to offer suggestions to improve them. The

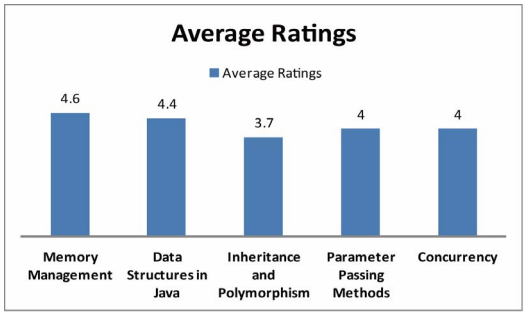

Figure 6.
Average ratings of learning objects grouped by topic.

Table 3.
The average ratings of the learning objects.

Learning Object	Ratings (5-point scale)
Memory management basics	4.9
Arrays in Java and C++	5.0
Static and Automatic variables	4.5
Information Hiding	3.9
Stack	4.4
Queue	3.8
Singly Linked List	4.9
Doubly Linked list	4.3
Inheritance concepts	3.8
Diamond inheritance problem	3.5
Polymorphism concepts	3.7
Parameter Passing Methods	4.0
Bounded Buffer Problem	4.3
Semaphores	4.0
Bound Buffer using Semaphores	3.7
Overall Mean	**4.18**

instrument used to gather these responses was an open-ended questionnaire, which was presented to the students with each learning object in addition to the request for a rating. When the students completed and submitted the evaluation, it appeared in the comments associated with the learning object in addition to the ratings. The other students could see the comments, which helped give them a sense of the usefulness of each learning object.

A thematic analysis was conducted to develop a list of categories to describe the students' perceptions of using the learning objects as learning aids. The responses are categorized based on the features that the students found most useful among the learning objects. Each category is supported by examples from the students' responses (**Table 5**).

Discussion

This study demonstrates that designing computer science learning objects based on different learning styles enhances the learning process by increasing students' motivation to use the learning materials. First, the study revealed that the students in the experimental course had diverse learning styles. The major-

Table 4.
Students' responses to the perception questionnaire.

Questions	Mean	SD
The animations used to describe different programming languages concepts were useful to me.	6.00	0.94
The learning objects were easy to understand.	5.89	1.15
The step-by-step descriptions of the concepts in the animation were helpful.	5.95	1.08
Highlighting the code during the animation helped me follow the code animation.	6.37	0.83
The text-to-speech sound used to describe the dynamic animation was useful.	3.89	1.10
The written description (at the bottom of the animation) helped me understand the dynamic animation.	6.16	0.83
The user controls at the bottom (show code, step-by-step, Java vs. C++, etc.) were helpful.	5.79	0.86
The learning objects that used examples from the real world to describe the concepts helped me grasp the concepts quickly.	6.05	0.91
The comments provided by other students helped me use the learning objects more efficiently.	5.74	1.24
Overall	**5.76**	**0.37**

Table 5.
Qualitative comment categories based on features of interest.

Feature of Interest	Supported examples from students' responses
Step-by-step description of the dynamic process	• "The ability to step through each process of a concept with an explanation and visual display" • "Everything is clearly explained step by step" • "The flow of information is what I like the most"
Highlighting of interesting events	• "Highlighting specific components to show their isolation" • "Highlighting and shadowing allowed consistent illustrations of the ideas and sound; much better and easier than drawing on a board"
Visual representation of abstract concepts	• "I think the visual animations were very nice" • "Easier to see what is going on" • "Yes. The visual animations do help with understanding different concepts. It is good to see how the data are stored"
The user's ability to control the animation	• "the implementation of student controls were also useful ideas for self-learning" • "changing the speed of the animation is useful for learning at my own pace"
Textual description of each step in the animation	• "The descriptions of what was happening were also very helpful" • "The description at the bottom of what is happening was also useful"

ity of students were visual learners (83%), compared to the 17% who were verbal learners. In addition, the majority of students were sensing learners (67%) who preferred concrete examples over abstract concepts, compared to 33% were intuitive learners with a preference for abstract concepts.

The average ratings of the learning objects indicated that students have positive perceptions toward using these learning objects to learn about different programming language concepts. This is reflected by the high overall average rating, which reaches 4.18 out of 5. Grouping the ratings based on the different topics covered in the course shows that the learning objects related to memory management concepts received the highest average ratings (4.6), while the learning objects related to the concepts of inheritance and polymorphism received the lowest average ratings (3.7). When learning about the memory management concepts of programming languages, it is essential for students to see how the data are stored in different sections of memory. The learning object designed for this study used animations to depict where different variables are stored in the memory and the dynamic changes in the memory as the program executes. Furthermore, the learning objects presented a comparison between the memory management approaches of Java and C++. The high ratings assigned to the memory management learning objects indicated that using the animation to design these learning objects is educationally effective. The students identified the features of the learning objects that were the most useful to them. These include showing the dynamic changes in the data structure using animations accompanied by highlighting of the code at each step and including text descriptions during the animation. The results of the students' perception questionnaire reveal that the students perceived the learning objects as easy to use and useful to support their self-regulated learning.

Conclusion and Future Work

The main objective of the research presented in this paper was to design learning objects to support students in their self-regulated learning of computer science. The study proposed a framework based on the theory of learning styles to improve the design of and interactions with learning objects. Based on this framework, a number of learning objects were designed to support the students' learning of certain programming language concepts. All the learning objects were stored in an online collaborative repository that identified the students' preferred learning styles and monitored the students' interactions with the learning objects. The study revealed that the students in the course had diverse learning styles. The majority of students were visual and sensing learners. The result of using these learning objects during the course revealed that the students perceived the learning objects to be easy to use and useful in supporting their learning about programming language concepts.

It should be noted that the sample size was not very large and based on this, the result of the study may be valid only in the area of teaching programming languages. However, the study used mixed methods of data collection which increases the validity and the generalization of the results.

The work described in this paper will continue by comparing the final exam scores of the students who used the learning objects this semester with those of another group who were taught using the traditional instructional approach and did not use the learning objects. In addition, an analysis of the students' interactions with the learning objects will be conducted. This will help obtain further insight into the students' self-regulated learning behaviors.

REFERENCES

Akdemir, O., & Koszalka, T. (2008). Investigating the relationships among instructional strategies and learning styles in online environments. *Computers & Education, 50*, 1451-1461. doi:10.1016/j.compedu.2007.01.004

Alharbi, A., Paul, D., Henskens, F., & Hannaford, M. (2011). An investigation into the learning styles and self-regulated learning strategies for computer science students. In *Proceedings Ascilite 2011* (pp. 36-46). Hobart: Ascilite.

Allert, J. (2004). Learning style and factors contributing to success in an introductory computer science course. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies* (pp. 385-389). Charlotte: SIGCSE. doi:10.1109/ICALT.2004.1357442

Ben-Ari, M., Berglund, A., Booth, S., & Holmboe, C. (2004). What do we mean by theoretically sound research in computer science education? *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 230-231). New York, NY: ACM.

Benaya, T., & Zur, E. (2008). Understanding object oriented programming concepts in an advanced programming course. *Informatics Education—Supporting Computational Thinking, 5090*, 161-170. doi:10.1007/978-3-540-69924-8_15

Berglund, A., Eckerdal, A., Pears, A., East, P., Kinnunen, P., Malmi, L. et al. (2009). Learning computer science: Perceptions, actions and roles. *European Journal of Engineering Education, 34*, 327-338. doi:10.1080/03043790902989168

Bruning, R., Schraw, G., & Ronning, R. (1999). *Cognitive psychology and instruction* (3rd ed.). Upper Saddle River, NJ: Prentice-Hall.

Chamillard, A., & Karolick, D. (1999). Using learning style data in an introductory computer science course. *ACM SIGCSE Bulletin, 31*, 291-295. doi:10.1145/384266.299790

Coffield, F. (2004). *Learning styles and pedagogy in post-16 learning: A systematic and critical review*. London: Learning and Skills Research Centre.

Coffield, F., Moseley, D., Hall, E., & Ecclestone, K. (2004). *Should we be using learning styles*. London: Learning and Skills Research Centre.

De Raadt, M., & Simon (2011). My students don't learn the way I do. *Proceedings of the 13th Australasian Computing Education Conference (ACE 2011)* (pp. 105-112). Perth: Australian Computer Society, Inc.

Dettori, G., & Persico, D. (2008). Detecting self-regulated learning in online communities by means of interaction analysis. *IEEE Transactions on Learning Technologies, 1*, 11-19. doi:10.1109/TLT.2008.7

Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. *Engineering Education, 78*, 674-681.

Felder, R. M., & Soloman, B. A. (1997). Index of learning styles. URL (last checked 20 June 2011). http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSpage.html

Fjuk, A., Bennedsen, J., Berge, O., & Caspersen, M. E. (2004). Learning object-orientation through ICT-mediated apprenticeship. *Proceedings of IEEE International Conference on Advanced Learning Technologies* (pp. 380-384). Washington, DC: IEEE Computer Society. doi:10.1109/ICALT.2004.1357441

Fleury, A. E. (2001). Encapsualtion and reuse as viewed by java students. URL. http://dis.eafit.edu.co/depto/documentos/p189-fleury%20-%20Encapsulation%20and%20Reuse%20as%20Viewed%20by%20Java%20Students.pdf

Hadar, I., & Leron, U. (2008). How intuitive is object-oriented design? *Communications of the ACM, 51*, 41-46.

doi:10.1145/1342327.1342336
Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding object misconceptions. *ACM SIGCSE Bulletin, 29,* 131-134. doi:10.1145/268085.268132
Holmboe, C., McIver, L., & George, C. (2001). Research agenda for computer science education. In *Proceedings of PPIG* (pp. 207-223). Bournemouth: 13th Workshop of the Psychology of Programming Interest Group.
IEEE/ACM (2005). Computing curricula 2005: The overview report. URL (last checked 20 June 2011). http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf
Keefe, J. W. (1988). Profiling and utilizing learning style. Reston, VA: NASSP Learning Style Series.
Liberman, N., Beeri, C., & Ben-David Kolikant, Y. (2011). Difficulties in learning inheritance and polymorphism. *ACM Transactions on Computing Education (TOCE), 11,* 4. doi:10.1145/1921607.1921611
LTSC (2002). Draft standard for learning object metadata. URL (last checked 4 July 2011). http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
Mills, J., Ayre, M., Hands, D., & Carden, P. (2010). Learning about learning styles: Can it improve engineering education? *Mountain Rise, 2,* 16.
Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—Views of students and tutors. *Education and Information Technologies, 7,* 55-66. doi:10.1023/A:1015362608943
Or-Bach, R., & Lavy, I. (2004). Cognitive activities of abstraction in object orientation: An empirical study. *ACM SIGCSE Bulletin, 36,* 82-86. doi:10.1145/1024338.1024378
Pintrich, P., & Zusho, A. (2007). Student motivation and self-regulated learning in the college classroom. *The Scholarship of Teaching and Learning in Higher Education: An Evidence-Based Perspective, 3,* 731-810. doi:10.1007/1-4020-5742-3_16
Pritchard, A. (2009). *Ways of learning: Learning theories and learning styles in the classroom* (2nd ed.). London: David Fulton Publishers.
Ragonis, N., & Ben-Ari, M. (2005a). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education, 15,* 203-221
Ragonis, N., & Ben-Ari, M. (2005b). On understanding the statics and dynamics of object-oriented programs. *ACM SIGCSE Bulletin, 37,* 226-230.
Reigeluth, C. M. (1999). What is instructional-design theory and how is it changing. *Instructional-Design Theories and Models, 2,* 5-29.
Sahami, M., Guzdial, M., McGettrick, A., & Roach, S. (2011). Setting the stage for computing curricula 2013: Computer science—Report from the ACM/IEEE-CS joint task force. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 161-162). New York: ACM.
Sosteric, M., & Hesemeier, S. (2002). When is a learning object not an object: A first step towards a theory of learning objects. *The International Review of Research in Open and Distance Learning, 3,* 2.
Thagard (Ed.) (2010). *The Stanford encyclopedia of philosophy*. Palo Alto, CA: Stanford University.
Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. *ACM SIGCSE Bulletin, 34,* 33-37. doi:10.1145/563517.563352
Watson, J. (1997). *Behaviorism*. New Brunswick, NJ: Transaction Publishers.
Weinstein, C. E., & Acee, T. W. (2008). Cognitive view of learning. In N. Salkind, & K. Rasmussen (Eds.), *Encyclopedia of educational psychology*. New York: Sage Publications.
Wiley, D. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. *Learning Technology, 2830,* 1-35.
Wiley, D. (2002). Learning objects need instructional design theory. In A. Rossett (Ed.), *The ASTD e-Learning Handbook* (pp. 115-126). New York, NY: McGraw-Hill.
Wiley, D., Waters, S., Dawson, D., Lambert, B., Barclay, M., Wade, D., et al. (2004). Overcoming the limitations of learning objects. *Journal of Educational Multimedia and Hypermedia, 13,* 507-521.
Zander, C., Thomas, L., Simon, B., Murphy, L., McCauley, R., Hanks, B. et al. (2009). Learning styles: Novices decide. *ACM SIGCSE Bulletin, 41,* 223-227.

