2014

A Cognitive Apprenticeship Approach for Teaching Abstract and Complex Skills in an Online Learning Environment

Reinaldo Fernandez
Nova Southeastern University, reinaldo@nova.edu

A Cognitive Apprenticeship Approach for Teaching Abstract and Complex Skills in an Online Learning Environment

by

Reinaldo D. Fernandez

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computing Technology in Education

Graduate School of Computer and Information Sciences
Nova Southeastern University

July 19, 2014

We hereby certify that this dissertation, submitted by Reinaldo Fernandez, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

_____ _____

Martha M. Snyder, Ph.D. Date
Chairperson of Dissertation Committee

_____ _____

Kellie W. Price, Ph.D Date
Dissertation Committee Member

_____ _____

Gertrude Abramson, Ed.D. Date
Dissertation Committee Member

Approved:

_____ _____

Eric S. Ackerman, Ph.D. Date
Dean, Graduate School of Computer and Information Sciences

Graduate School of Computer and Information Sciences
Nova Southeastern University

2014

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

A Cognitive Apprenticeship Approach for Teaching Abstract and Complex Skills in an Online Learning Environment

by
Reinaldo Fernandez
July 2014

Undergraduate courses such as mathematics, science, and computer programming require high levels of decision making, concentration, and cognitive demand. Researchers in the field of instructional design are interested in effective instructional strategies that can aid practitioners in teaching such abstract and complex skills.

One example of an instructional strategy that has proven effective in teaching these skills is cognitive apprenticeship (CA). While CA has been applied to courses such as mathematics and computer programming in face-to-face and blended learning environments, there is little evidence of the advantages of applying CA in a fully online computer programming course. Specifically, the introductory programming course, CS1, is the first contact that undergraduate computer science students have with their chosen major. Historically, drop-out rates for CS1 have been high and thus strategies for effective teaching of this course have served as an important topic in the research literature.

The goal was to design and validate internally an online CS1 course that incorporates CA strategies. A two-phase design and development research method was used to guide the construction and internal validation of a fully online CS1 course. Phase one resulted in the design and development of the course guide. An expert-review process using the Delphi technique was implemented in phase two to validate the design with regard to its effectiveness, efficiency, and appeal. Three rounds of review by the panel resulted in consensus.

Results from the expert-review confirmed the application of CA as an effective, efficient, and appealing instructional strategy to use when designing an online CS1 course. Future research should focus on external validation of the design by implementing the course to evaluate its effectiveness, efficiency, and appeal among stakeholders. In addition, it is hoped that the course guide can be used to help practitioners design and implement a fully online CS1 course that uses CA strategies.

Acknowledgments

I would like to express my deepest appreciation to my committee chair Professor Martha Snyder, who has been an indefatigable support throughout the entire dissertation process. I am sure that without her guidance and leadership, this effort would not have been possible.

I would also like to thank my committee members, Professor Trudy Abramson and Professor Kellie Price, whose guidance and ready disposition to help led me to the successful completion of this work.

A great word of thanks goes to the Delphi panel members --who by design must remain anonymous-- that so graciously participated in this study. Their dedication to optimizing the course design over several months of going back and forth with revisions is a true testament to their stamina and desire for excellence.

Last but not least, I would also like to express my most sincere thanks to my best friend and lifelong companion, my wife Raquel. Without her support when I was down it would not have been possible to finish this dissertation.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter One

Introduction

Background

Computer programming is an abstract and complex skill which requires persistent practice in order to learn it well. Learning to program a computer is a famously difficult task, with a very prolific research literature dedicated to studying it (Valentine, 2004). Like similar subjects including math and science, computer programming can be difficult because of the abstract and highly complex skills that are involved.

In 1978, the curriculum committee of the Association for Computing Machinery (ACM) published recommendations for undergraduate computer science programs (Austing, Barnes, Bonnette, Engel, & Stokes, 1979). This curriculum design has been updated through the years and in the 2001 revision (ACM, 2001), the committee identified one course, CS111, also widely known in the literature as simply "CS1," as the first course that all computer science majors were required to take. After all these years, there is little agreement regarding the exact content to be covered in the CS1 course, although it typically covers basic programming skills and basic data structures. According to the 2001 curriculum design, CS1 covers the following specific items (in the imperative-first, traditional topic-based approach): (1) fundamental programming constructs, (2) algorithms and problem-solving, (3) fundamental data structures, (4) fundamental computing algorithms, (5) basic computability, (6) assembly-level machine organization, (7) overview of programming languages, (8) declarations and types, (9) abstraction mechanisms, (10) object-oriented programming, (11) fundamental techniques in graphics, (12) history of computing, (13)

software design, (14) software tools and environments, (15) software requirements and specifications, and (16) software validation.

For many students, CS1 is difficult and often students give up and abandon computer science for other disciplines. Historically, drop-out rates for CS1 have been high and thus, strategies for effective teaching of this course have served as an important topic in the research literature for many years (Valentine, 2004). In fact, Kumar (2013) notes that the ACM special interest group in computer science education, or SIGCSE, is often nicknamed among practitioners as "SIGCS1" due to the prolific output of researchers studying that important first course. Based on a world-wide survey, Bennedsen and Caspersen (2007) determined that CS1 has an average failure rate of 33%. Compared to the 26.8% college failure rate overall, the CS1 failure rate is 6.2 percentage points, or 23%, higher. Given the number of students enrolled in computer science (two million at that time), a one percentage point improvement would equate to 20,000 additional students successfully finishing their studies. The results of this survey underscore the strong negative impact of the CS1 problem. When CS1 is taken online, the failure rate is even higher, sometimes reaching as high as 50% (Vilner, Zur, & Sagi, 2012). The problem with CS1 is primarily the result of the cognitive complexity that many students face while learning the difficult skill of computer programming, as analyzed by Robins, Rountree, and Rountree (2003).

One effective strategy that has shown promise in face-to-face and blended learning environments for teaching computer science is cognitive apprenticeship (CA) (Mow, Wing, & Yates, 2006; Vihavainen, Paksula & Luukkainen, 2011; Knobelsdorf, Kreitz, and Bohne, 2014). CA is an instructional strategy comprised of six instructional methods: modeling, coaching, scaffolding, articulation, reflection, and exploration (Collins, Brown, & Newman, 1989). By

modeling, the expert shows the student the thinking process needed to develop the solution to a problem. *Coaching* refers to the student working on problems while the expert offers guidance in applying the correct approach to the solution. *Scaffolding* refers to the gradual removal of the expert's guidance so that the student can develop more and more of the problems independently. *Articulation* refers to the requirement that students explain why a specific approach to a problem works. *Reflection* entails comparing and criticizing the student's work with other students or the instructor, with the intent of developing a producer-critic internal dialogue. *Exploration* refers to trying out new approaches to resolution of the problems, with the intent of developing independent thought (Kuo, Hwang, Chen, & Chen, 2012). The strategy melds apprenticeship concepts and traditional schooling aimed at teaching and learning of cognitive skills. The idea is to "make the target processes [or thinking] visible" (Collins, et al., 1991), or to clearly present to students the thinking process that an expert in the subject area develops when resolving problems in the chosen domain. The first three instructional methods (modeling, coaching, and scaffolding) were part of the traditional apprenticeship model (Collins, Brown, & Holum, 1991). The three additional elements to the conceptual framework: articulation, reflection, and exploration, are aimed at externalizing thinking processes that are usually internal, not explicitly explained. Once this thinking is made visible, it can be more easily reproduced and in fact, the thinking process itself is learned.

Problem Statement and Goal

The acquisition of programming skills is difficult because of its cognitive complexity (Robins, Rountree, & Rountree, 2003). When these skills are taught in an online classroom, the

difficulty of the learning task is compounded due to the difficulty inherent in offering the immediate, hands-on guidance and scaffolding that is often needed to teach these skills.

Cognitive apprenticeship (CA) (Collins, et al., 1989) is an instructional strategy that has proven successful in teaching courses in abstract and complex topics such as mathematics and science. While CA has been applied to these types of courses in face-to-face and blended learning environments, there is little indication in the literature on how to apply CA instructional methods in fully online CS1 courses. Ramdass (2012), for example, reviewed the literature and discussed the application of CA in online science courses, reviewing several approaches that have been used for that. One of these CA-based approaches is using science-based computer games as a learner-centric learning model, and they have helped develop students' interest in science-related careers, as detailed by Beier, Miller, and Wang (2012) and also by Foster (2008). Ramdass also presented another such approach (Mayer, Mautone, & Prothero, 2002) that used gaming-style simulation of the terrain in a given area and had students solve geology problems through virtual surveying and analysis of the virtual terrain's features. A third approach presented by Ramdass (Tsang, Naughton, Leong, Hill, Kelly, & Leahy, 2008) offered medical training to students through virtual reality simulations of patient interactions, evaluated by a panel of experts. Only when a level of training was mastered were students allowed to step up to the next level of training.

In particular, scaffolding is the feature that appears more difficult to adapt to a fully online course. Three basic approaches to scaffolding have been found in the literature: a face-to-face approach, a question and peer collaboration approach, and an automated tutor approach. First, Vihavainen, Paksula, Luukkainen, and Kurhila (2011) implemented a face-to-face scaffolding approach, which required the support of several instructors and a group of mid-level

and advanced students of computer science. The ratio of guide to student was 1:10. The exercises were developed in a lab environment with both students and guides in attendance. There were a total of 20 hours per week of such sessions throughout the complete 14-week course. The exercises were evaluated on the spot by the guides as exercises were finished by students. Exercises were returned by the guide and re-worked by the student until the exercise had been developed with a good level of quality. Second, Ge and Land (2004) presented a scaffolding approach applied to complex, ill-structured problem learning in various learning environments, including online. This approach consisted of question prompts and peer collaboration to discuss the questions. The instructor, who played the role of subject matter expert, guided the discussion. The style and substance of the scaffolding questions suggested in this study are aligned with the IDEAL problem-solving learning strategy proposed by Bransford and Stein (1993). Incidentally, this IDEAL approach seems particularly well suited to an online CS1 course, where this technique could be implemented through a discussion forum where the questions are posted for discussion and the programming exercises are analyzed in depth. Third, in a massive open online course (MOOC)-style CS1 course developed by Vihavainen, Luukkainen, and Kurhila, (2012) a scaffolding approach was implemented based on an automated tutoring software. The automated tutor contained a set of exercises with increasing difficulty levels that were automatically graded by the tutor software and which offered tips and suggestions about how to solve difficult problems when the automated tutor noticed that a student struggled with an exercise. The program automatically advanced students into more complex exercises, while gradually reducing the guidance so that proper scaffolding was implemented.

All three of these basic approaches to implement scaffolding have resulted in encouraging results, although varied. In the first approach, face-to-face scaffolding (Vihavainen,

et al., 2011), the results indicated that there was an improvement in student performance from 51.1% to 70.7% for the first part of the course, and from 52.7% to 82% for the second part of the course. The second scaffolding approach was implemented by Demetriadis, Papadopoulos, Stamelos, and Fischer (2008) in a computer software project management course and the researchers found statistically significant improvements in class performance. In the study describing the third approach, fully online scaffolding (Vihavainen, et al., 2012), the researchers reported positive anecdotal survey feedback from students, pointing to both the effectiveness and toughness of the course as designed. While CA has been applied to courses such as mathematics (Collins, et al., 1989), science, technology, and computer programming in face-to-face and blended learning environments, there is little evidence of the advantages of applying CA in a fully online computer programming course.

The goal was to design and validate internally an online CS1 course guide that incorporates CA strategies. When describing instructional design model validation, Richey and Klein (2007) defined internal validation as the "empirical verification of the components and processes included in a design and development model" (p. 158). In the case of internal validation of an online CS1 course design, the aim was to validate the application of CA strategies to teach various facts, concepts, principles, rules and procedures in a fully online course. When considering the validation of instructional design theories, models, and methods, a major concern is *preferability*. Reigeluth and Frick (1999) define preferability as "the extent to which a method is better than other known methods for attaining the desired outcome" (p. 634). That is, what instructional methods are most preferable and in which situations? Organized into three dimensions (i.e., effectiveness, efficiency, and appeal), the authors explain how these three criteria can be used to generate design knowledge. However, they warn that these criteria are

sensitive to the context of the instruction so the needs and wants of the stakeholders should be considered. There may be some instances where effectiveness is most important while other instances value appeal as the most important criteria. In the context of a course, Doering and Veletsianos (2008) define *effectiveness* as the ability to comply with the stated learning goals established for it. *Efficiency* is achieved when effectiveness is accomplished using the least amount of resources possible. *Appeal* refers to the willingness of students to become engaged and immersed in the learning experience and enjoy the learning process. These three criteria can be used to evaluate whether specific instructional methods work within a specific context. Therefore, these criteria are incorporated into the validation of the CS1 course design.

Research Questions

The following research questions guided this investigation:

RQ1. What is the evidence that specific elements of CA are helpful to students in an online CS1 and similar online courses that require abstract and complex thinking?

RQ2. How must the traditional course be revised to incorporate the selected CA elements for effective online delivery?

RQ3. What are the reactions of experienced instructors of CS1 to the proposed CA-supported online course in terms of effectiveness, efficiency, and appeal?

RQ4: What are the modifications needed to improve the proposed CA-supported CS1 online course in terms of perceived effectiveness, efficiency, and appeal?

To answer the first research question, a thorough review of the literature was conducted. The review focused on studies pertaining to the design and development of courses that required abstract and complex thinking and the instructional theories and methods, including CA methods that were used to support these designs in both face-to-face and fully online courses. The second research question was answered by using a systematic instructional design process to guide the development of the CA-supported CS1 course (Appendix A). This process included the selection of appropriate CA methods that support the acquisition of the knowledge and skills required for the online CS1 course. The third research question was answered through the solicitation and collection of input from experienced instructors who teach CS1 online. Special attention was given to the elements of effectiveness, efficiency, and appeal of the instructional design. Finally, research question four was answered by analyzing the data and using it to make modifications to the proposed CS1 course design.

Relevance and Significance

The difficulties that students have with the first encounter with a computer programming course such as in the CS1 course have been investigated in detail (e.g., Herrmann, Popyack, Char, & Zoski, 2004; Hughes & Ramanee-Peiris, 2006). Different approaches have been implemented to improve such courses with varying degrees of success. Most of the studies focused on face-to-face or blended learning environments, but very little attention has been given

to the online CS1 course. Given the increase in the delivery of fully online courses in higher education, including courses such as mathematics, science, and computer science, it becomes increasingly relevant and useful to analyze instructional strategies that will improve performance and retention in such courses. As such, focusing on this particular mode of delivery offers practitioners who teach CS1 courses in a fully online environment, guidelines for implementing CA strategies. More broadly, however, while this investigation focused solely on computer science and programming skills in particular, the design could be applied to other fully online courses where the development of abstract and complex skills are required.

Barriers and Issues

The difficulties many students encounter when learning computer programming for the first time have been studied in the academic community for many years. A multitude of studies have been developed over several decades attempting one solution or another (Pears, Seidman, Malmi, Mannila, Adams, Bennedsen, Devlin, & Paterson, 2007), and CA has merely been one of the many alternate approaches attempted, often with good results (Mow, Wing, & Yates, 2006; Hassinen, & Mayra, 2006). Nearly all of these studies, however, have addressed solutions to this problem in the face-to-face or blended classroom, but few studies have been developed analyzing solutions to this problem in a fully online classroom. Designing a CS1 course for a fully online delivery is inherently difficult and as such requires careful thought in the instructional design of the course activities in that they must be effective in achieving the stated learning objectives, efficient in the use of instructor and student time, and appealing to all stakeholders involved (i.e., students, faculty, and administrators). In addition to this inherent difficulty, there are other issues with respect to the context that were used to guide the design of the CS1 course. The researcher

currently teaches an online CS1 course at a for-profit educational institution that uses the eCollege learning management system (Pearson eCollege, 2013). Even though the scope of this study did not include the implementation of the CA-modified course, the design was based on the affordances and constraints that exist at the institution. Following is a brief description of the context.

Each course is designed by the centralized instructional design department of the institution and consists of (per week of instruction): (1) written lecture, (2) live lecture (videoconferencing), (3) two discussion forums, (4) one quiz, and (5) one programming problem. In addition, a final exam and a textbook are included. Instructors do not have full academic freedom as defined by the AAUP (2006; paragraph 2 of the section on Academic Freedom) and are instead contractually required to follow the pre-established syllabus. As such, instructors cannot change the assessment components of the course (quizzes, final exam, programming problems, number of discussion forums, and the points assigned to each of these items). On the other hand, instructors are able to change the discussion questions (but not the points assessed). Instructors are also able and encouraged to add individualized content, such as additional written lectures; additional discussion questions (provided that the overall number of points assigned to discussions is not changed); videoconferencing sessions with students, individual or in groups, (although participation in these sessions cannot be required – they must be optional for students); additional individual forums for one-on-one discussions between instructor and each student; modifications to the *description* of the programming problems, but without modifying the required program itself, which must remain as centrally developed by the institution; and text-only "chat" sessions with students. Also, instructors can offer grading feedback for the weekly programming problems in any way desired, provided it is based on a grading rubric (which is not

centrally developed by the institution but by each professor). Such rubric-based grading feedback requires that the instructor not only jots down the points earned in each assessment item in the rubric but also adds written comments that point out opportunities for improvement in the student's work and the correct way to solve or implement each specific issue, if needed.

The course design complied with these requirements and the instructor was contractually required not to deviate from them except as indicated. Due to this situation, weaving the components of CA into this course, particularly the scaffolding component, was particularly difficult. The other components of CA, namely modeling, coaching, articulation, reflection, and exploration were easier to incorporate given the academic freedom constraints of the target institution. There were, however, other complications. As noted, the target institution did not allow changes to the standard course syllabus, particularly with respect to the assessments (points assigned to each of the course's components). This meant that any additional activity or assignment is optional. That is, students are not required to complete it. The author's experience in this target institution indicates that without an assessment incentive, few students take advantage of any added feature, no matter how convenient to their learning it may appear to be. This issue complicated the instructional design for some of the components of CA, namely articulation and reflection.

How generalized are the academic freedom restrictions of the target institution? As part of an analysis of for-profit universities such as the target institution, Tierney (2011) explained that faculty life is significantly different in such institutions when compared with traditional universities. Of note is the fact that in such traditional institutions the instructor prepares the syllabus according to what is perceived as important by the individual instructor, whereas in the for-profit universities, the syllabus is centrally developed and modifications are, in general, not

allowed. Morey (2004) also explained this marked difference between the traditional university and the for-profit university, where instructional delivery is independent of instructional design.

Definitions and Acronyms

ACM – Association for Computing Machinery.

Appeal (in instructional design) – Refers to the willingness of students to become engaged and immersed in the learning experience and enjoy the learning process (Doering &Veletsianos, 2008).

Articulation (cognitive apprenticeship) – Refers to the requirement that students explain why a specific approach to a problem works (Collins, Brown, & Newman, 1989).

Blended classroom – Classroom merging the traditional, face-to-face classroom with an online classroom, typically implemented through a limited number of face-to-face meetings scheduled throughout the course's duration plus more frequent, often daily, online interactions through a learning management system (Rosenberg, 2003).

CA – cognitive apprenticeship (Collins, et al., 1989).

Coaching (cognitive apprenticeship) – Refers to the student working on problems while the expert offers guidance in applying the correct approach to the solution (Collins, Brown, & Newman, 1989).

CS1 – Computer Science "1" course, or first course in computer science programs, covering computer programming and basic data structures as main topics (Bowles, 1978).

Effectiveness (in instructional design) – Refers to the ability of an instructional design to comply with the stated learning goals established for it (Doering &Veletsianos, 2008).

Efficiency (in instructional design) – Is achieved when effectiveness is accomplished using the least amount of resources possible (Doering & Veletsianos, 2008).

Exploration (cognitive apprenticeship) – Trying out new approaches to resolution of the problems, with the intent of developing independent thought (Kuo, Hwang, Chen, & Chen, 2012).

Face-to-face classroom – Traditional, lecture-based classroom, where the instructor imparts lectures to students seated at chairs or desks (Toff, 1955).

Modeling (cognitive apprenticeship) – The expert shows the student the thinking process needed to develop the solution to a problem (Collins, Brown, & Newman, 1989).

Online classroom – Computer-mediated classroom implemented through a Learning Management System such as eCollege (Pearson eCollege, 2013).

Reflection (cognitive apprenticeship) – Self-analysis of the student's problem-solving approach, comparing the student's approach to the expert's approach (Collins, Brown, & Newman, 1989.)

Scaffolding (cognitive apprenticeship) – Gradual removal of the expert's guidance so that the student can develop more and more of the problem solutions independently (Collins, Brown, & Newman, 1989).

Summary

Due to its abstract and complex nature, the first computer programming course is a major stumbling block in the computer science studies of many students. With varying degrees of success, instructional strategies have been implemented in an attempt to help students learn the material in an effective, efficient, and appealing way. One such strategy which shows great promise is CA.

Given the rapid increase in the delivery of online learning in higher education, it has become increasingly relevant to identify instructional strategies that can effectively facilitate the acquisition of abstract and complex skills. By constructing a fully online CS1 course guided by CA and validating the design, steps are being taken toward the development of effective, efficient, and appealing strategies for redesigning online courses that require abstract and complex skills.

The following chapter includes a review of the literature related to teaching abstract and complex skills including an overview of the CS1 course in particular. Examples of CS1 solutions are surveyed and organized into four categories; front-end filtering, improved teaching and learning aids, live instruction, and CA. Next, research related to CA and its application to CS1 is discussed including specific principles associated with CA as an instructional strategy including constructivism, zone of proximal development, scaffolding, situated cognition, cognitive load theory, and problem-based learning. The design and development research approach (Richey & Klein, 2007) is outlined in Chapter 3. Two overarching phases – development and validation – are described along with the detailed procedures associated with each phase. Chapter 4 will present the Results obtained in the study and Chapter 5 will round up the study with conclusions, implications, and recommendations.

Chapter Two

Review of the Literature

This review of the literature highlights existing research pertaining to the investigation. This research includes: teaching abstract and complex skills, the first computer science course (CS1), examples of CS1 solutions, cognitive apprenticeship (CA) as an instructional strategy, and the affordances of CA as an instructional strategy in the design of a fully online CS1 course.

Teaching Abstract and Complex Skills

Computer programming is a complex skill. The intrinsic cognitive load placed on an individual when learning how to program is high. As a result, researchers in the field of instructional design are interested in how computer programming tasks can be broken down in a way that lessens cognitive load and promotes learning. In an extensive description and review of cognitive load theory, Von Merrienboer and Sweller (2005) explained some of the instructional methods developed in consideration of the theory. The essence of the theory has a biological underpinning: human memory is divided into long-term memory and short-term memory. All learning is acquired through short-term memory and then classified in a permanent way into long-term memory in a second phase, at most 20 seconds or so later, and is stored as relational schemas that can be recalled when needed as a group. Working memory can hold a maximum of seven elements but it can operate on only two to four at the same time. This heuristic works for new data received from the outside but for internally-sourced data (i.e., schemas recalled from long-term memory), there is no known limit to the number of elements that can be handled simultaneously. As a consequence, when designing instruction, an effort should be made to

reduce the number of simultaneous knowledge bits presented to the learner, and as such the creation of the necessary long-term memory schemas will be enhanced.

Garcia, Sánchez, and Acuña (2013) analyzed complex, multimedia learning activities (within the topic of plate tectonics) and sought to determine if learners do better with broad support systems or if they do as well with less intense support systems. When facing such complex learning activities, self-regulation of the learning process is necessary (that is, planning, monitoring, and regulating both cognition and motivation). The authors used an experimental approach. Eighty-nine undergraduate students were randomly assigned to four groups with: (1) minimal support (planning support), (2) intermediate support, with two subgroups, (2.a) planning support and corrective feedback on questions and (2.b) planning support and explanatory feedback, and (3) broad support, with planning support, corrective feedback on questions, and explanatory feedback. Students were tested before and after the learning event, with an independent expert grading the tests. A detailed statistical analysis indicated that minimal to intermediate levels of support led to significantly less effective learning than a broad level of support. One consequence of the results was that learners with low prior knowledge of a topic found difficulties when self-regulating their learning process, which in turn indicated that strong instructor's support was needed. The authors also noted that in order for learners to acquire the needed self-regulation knowledge, so they can be independent of support in the future, scaffolding should be applied to the level of instructional support.

Specifically, CS1 presents difficulties to students due to the very high degree of complexity and abstraction that computer programming presents to novices, as mentioned by Rich, Perry, and Guzdial (2004). Of particular interest, Robins, Rountree, and Rountree (2003) stated that computer programming is a difficult, complex endeavor which parallels other

complex cognitive skills such as chess playing and mathematics. These researchers explained that learning to program entails five difficult topics, which represent various types of content (i.e., concepts, principles, rules, and procedures) that must be acquired: (1) orientation (i.e., what a program can do and what it is used for), (2) the notional machine (i.e., a mental representation of the computer as an executor of programs), (3) notation (i.e., the particular syntax of a computer language), (4) structures (i.e., algorithmic models for solving problems), and (5) pragmatics (i.e., the skills of analysis of a problem, design of a computerized, solution, debugging, and testing). The fact that these five topics and their respective content types must be juggled at the same time when attempting to successfully write a computer program, leads to a very high cognitive load.

In a subsequent study, however, Robins (2010) studied this specific topic of the difficulty of learning computer programming in great detail and explained that learning computer programming has an additional difficulty: the sequential nature of the learning topics and how they accumulate in time. That is, what students learn today will be used tomorrow in different, more complex ways, so failing to fully understand one specific topic will mean that future topics will be less understood. This compounded lack of knowledge and skill accumulates over the duration of the course and it is this characteristic of computer programming learning that results in what Robins called the "learning edge momentum" and which offered as result a bimodal distribution of grades (either very good or very bad grades, with few intermediate grades). The traditional but simplistic explanation for this bimodal distribution is that there are two types of people: programmers and non-programmers, which means that programming ability is somehow innate. In an extensive review of the literature, Robins found that there is no clear personal characteristic that can predict success in a first-time encounter with computer programming, with

the exception of general IQ – the higher the IQ the more probable the success in a first-time computer programming course. On the other hand, however, this does not explain the bimodal grade distribution, since the IQ distribution in the general population roughly follows a bell curve.

Continuing the discussion of Robins (2010), this bimodal distribution of grades appears to have no explanation, even though it has been solidly observed across multiple countries, computer languages, and instructors. Mathematically, the bimodal distribution of grades can be represented through a model where a course's learning is divided into "chunks," and the grade for each such chunk depends on the grade earned in the previous chunk. That is, if the grade in chunk "n" is below a certain value, then the threshold for earning a passing grade chunk "n +1" is raised; and the contrary, if chunk "n" passed with a good grade, then chunk "n + 1" has a lower threshold value. Such a model results in the observed anti-normal distribution of grades and which Robins called the momentum effect. This momentum effect can be understood by considering two ideas: (1) computer programming is composed of well-defined and closely-connected elements and (2) we learn by adding new knowledge to the schemas already stored in our long-term memory. That is, we learn at the boundaries of that which we already know, or at the zone of proximal development. Based on these ideas, then, the "learning edge momentum" when applied to CS1 means that successful learning of one topic will facilitate learning future topics and the contrary, that unsuccessful learning of a topic will result in less-successful learning of future topics. This idea makes learning self-reinforcing, negatively or positively, and thus the term "momentum." The observed bimodal distribution of grades occurs as a direct result of the topic of computer programming. Are these bimodal grade distributions unique to CS1?

Robins pointed out that no, courses in other complex topics such as statistics, languages, music, and science also show such grade distributions in varying degrees.

The Computer Science 1 Course

The first contact that undergraduate computer science students have with their chosen major is the computer science 1 (CS1) course (Bowles, 1978), which is designed to introduce the student to the essentials of the profession. The failure rates in CS1 are significantly higher than the average for other courses and such failures often lead to students' abandonment of computer science as a career at a time when the job market needs for computer scientists is very high (Barker, McDowell, & Kalahar, 2009). Any effort to reduce the failure rate in CS1 could lead to better prepared and more satisfied students that may be more committed to remain in computer science, graduate, and find employment in the computer science field.

Many students find the CS1 course challenging and some find it so difficult that they end up choosing a different program of study and abandon computer science altogether. In fact, the attrition rate for the CS1 course is exceptionally high when compared to other introductory college-level courses (Robins, 2010).

There are numerous studies dealing with this problem, such as Herrmann, Popyack, Char, and Zoski (2004), where the researchers reported on a CS1 course at Drexel University with a "DFW rate" (students that earn a grade of "D" or "F" or who withdraw from the course with a "W") of up to 50%. As another example, Hughes and Ramanee-Peiris (2006) complained about the difficulties students face with this initial course. Many similar studies are found throughout the literature, such as Wang, Dong, Li, Zhang, and He, (2012), Ambrosio, Moreira, Almeida, Franco, and Macedo (2011), Vilner, Zue, and Sagi (2012), and Stone and Clark (2011).

In the annual SIGCSE technical symposium (the ACM's Special Interest Group on Computer Science Education), the number of articles dealing with the difficulties (and opportunities for improvement) in CS1 had been increasing at least up until a 2004 meta-analysis by Valentine (2004). An informal count by the author of the number of papers dealing with CS1 since then shows that the number seems to be holding steady or increasing slightly from the 2004 levels.

To compound the problem even further, these high failure rates in CS1 lead to student frustration and low continuing enrollment in computer science, which in turn result in low graduation rates at a time when job openings are high. For instance, Turner, Albert, Turner, and Latour (2007) stated that student retention is a serious problem for computer science departments and argued that students leave the program due to both a frustration with programming and disillusionment with computer science as a career. The dropping enrollment rates of computer science students (a drop of more than 70% over the 2005-2006 period) coupled with an increase in open job positions have led many researchers to investigate the reasons for such a worrisome situation (Wortman & Rheingans, 2007). Barnes, Richter, Powell, Chaffin, and Godwin (2007) also explained that the enrollment and graduation rates in computer science are declining, even though the demand for computing jobs is rising, with the computer science attrition rates reported in the 30% to 40% range. In times of declining enrollment in computer science, every effort must be made to retain as many students as possible. The survival of computer science departments is at stake.

The problem with CS1 is primarily the result of the cognitive complexity that many students face while learning the difficult skill of computer programming, as analyzed by Robins,

Rountree, and Rountree (2003). When this situation is compounded by a fully online delivery of CS1, the problem of retention and diminished performance intensifies.

Examples of CS1 Solutions

Various alternatives have been proposed to address the difficulties inherent in courses requiring the development of abstract and complex skills. While not exhaustive, the following examples reflect the recent research literature particularly from the IEEE and ACM databases.

There are four broad categories of proposed solutions for increasing student success in CS1. The first focuses on altering the qualifications of the incoming student (also known as front-end filtering). Those lacking the requisite skill or ability may either be directed to a preparatory course (known as CS0 or CS-Zero course) or, in more extreme cases, advised to enter a different course or discipline altogether. The second category of solution has to do with individual activities the students can do on their own time, such as automated tutoring, using a database of pre-recorded micro-lectures, or writing "practice" programs. The third category involves *live* instruction. *Live* in this context refers to the real-time presence of an instructor, whether physically present as in a classroom or present online in real time through video conferencing technologies. A fourth category involves using CA. A description of these four categories follows.

Category 1: Front-end Filtering

Improved Advising. One strategy to improve the DFW rate of CS1 is to avoid enrolling students who have a higher probability of failure. If such a set of parameters could be identified, then academic advisors could attempt to steer students likely to fail away from CS1. Although CS1 is the first course in computer science, and is of interest mainly to CS majors, there are

many other types of students that take the course as an elective and these are precisely the students that are more at risk of not finishing CS1 successfully. Rountree, Rountree, Robins, and Hannah (2004) developed a decision tree analysis to help determine the combination of parameters that best predicts success or failure in CS1. The results were surprising. Failure was predicted by the following four combinations of parameters: (1) (not looking for an A) and (second or third year) and (background not science), (2) (Humanities background) and (not looking for either an A or B), (3) (Third year) and (no Mathematics background), and (4) (background not science) and (not in first year) and (under 25 years of age) and (no Mathematics background).

Success, on the other hand, was predicted by the following two combinations of parameters: (1) (looking for an A) and (not expecting the course to be "hard"), and (2) (looking for an A) and (age is 16 to 18). The parameter "students who are looking for an A" appeared in both success rules. As to failure, however, some common parameters included: non-science, non-mathematics background, and students in upper-level years. It appears that if academic advisors try to dissuade upper-classmen with a non-scientific or non-mathematical background from taking CS1 as an elective, the DFW rate of the course should decline.

CS0 Course**.** For over sixteen years, the computer science teaching community has realized that adding a pre-CS1 course—commonly referred to as "CS0" in the literature—is a valid strategy for improving student performance in CS1. The CS0 course does not cover a specific programming language, but instead focuses on the all-important algorithmic approach to problem-solving. Allan and Kolesar (1997) implemented such a course as a prerequisite for CS1, and results were significant. Students who had taken CS0 as a prerequisite for CS1 earned nearly twice as many "A" grades as those who did not take CS0. The researchers explained that up until

then, the typical way in which students were introduced to computer science was through computer programming but they noted that given the variability in students interested in such a course –some aimed at computer science as a career, while others were aimed at different careers such as business. Before applying the CS0 course, the success rate (C- or better in final grade) was relatively low compared with average results (only 52% completion rate in one specific semester). The background of the students also influenced the success rate. While only 22% of students without any previous experience in computers finished the course successfully, 50% of those with some computer exposure, even spreadsheet use and word processing, succeeded. By college, the success rates are also different: 30% for Business majors; 51% for Engineering majors; 22% for Science majors; and only 10% for other majors. As a reply to these problems, Allan and Kolesar (1997) designed what they called a "CS0" course – a prerequisite of CS1. The CS0 course was based on face-to-face lectures but using a Socratic approach. They accompanied these lectures with practice sessions in small groups, where the focus was on problem-solving techniques, even based on challenges apparently not directly related to computer programming, such as rearranging a given set of toothpicks in a certain number of moves. The results showed an increase in grade point average from 2.8 to 3.2 (a 14% positive change).

Dierbach, Taylor, Zhou, and Zimand (2005) proposed having students take an assessment test and based on the results, have them either go directly into CS1 or go first into CS0. Before the study was developed, the CS1 course in Dierbach et al.'s (2005) institution simply had as prerequisite a computer programming course and basic mathematical expertise. However, this did not warrant that students were uniformly well prepared for CS1. By implementing the assessment test, Dierbach et al. were able to appropriately steer students to either CS0 or CS1, and after five consecutive years of applying the assessment, the results showed that the

assessment that was designed did correctly distinguish programming reasoning ability based on the strong correlation between assessment grade and CS1 course grade. Second, the CS0 course improved the grade point average of students in CS1: 2.00 for students with no background; 2.59 for a previous course in computer programming (30% better); and 2.84 (42% better) for those having taken CS0 before.

Category 2: Improved Teaching and Learning Aids

Database of Learning Objects. Recording brief *micro-lectures* where the instructor explains basic programming topics appears to be an effective way to improve the performance of a CS1 course. In fact, Bennedsen and Caspersen (2005) proposed such a strategy and reported positive results. These micro-lectures or process recordings were recorded using software such as Camtasia (2013). The recording process requires the instructor to sit at the computer and record the spoken explanation at the same time that the topic is being presented on screen, often using the live interactive development environment of the programming language being learned. A modeling technique is used, so that the programming design process is revealed to students. What the researchers proposed was to have small, bite-sized micro-lectures where a single topic is explained, so that students could easily play them back as often as necessary to understand the ideas and techniques used. These micro-lectures formed a database of learning objects that was accessible to students at any time. Bennedsen and Caspersen prepared a full set of these short videos, together with a video recording of the standard classroom lectures as well. Students chose to watch the short videos overwhelmingly over the recorded lectures: 91% of views for the short videos and only 9% for the lectures. The researchers concluded from these data that students preferred the short videos and in fact, 58% of students replied in a survey that they had a

high or very high degree of satisfaction with the course (but no data from previous iterations of the course without the short videos were included).

Vincenti, Braman, and Hilberg (2012) presented their design of reusable learning objects used in a CS1 course to help comply with the different learning styles of students. This pilot study only covered one major topic in CS1, repetitive computations, and did not go beyond that since the intent was to analyze how beneficial the type of learning object would be and thus pave the way for further developments in the future toward implementing the learning objects in additional topics within CS1. The learning objects were divided into content modules, each covering one particular aspect of repetitive computations, and, in turn, each of these modules had five distinct learning units, aimed at a different learning style and offering different venues for reviewing the topic: (1) text-based explanation, (2) video lecture, (3) demonstration with animation, (4) interactive, hands-on demonstration, and (5) assessment. The pilot study only contained the interactive, hands-on demonstration and these demonstrations were presented to students as part of the standard lectures offered and then were released to students so they could practice with them on their own at home. After the practices, students were given a questionnaire and asked to take it with their impressions of the interactive demonstrations. Overall, results were positive and on a scale of 1 to 10 (low - high), students found using the learning objects as satisfying (6.7), stimulating (6.2), and easy (6.7). As to effectiveness, on a scale of 1 to 5 (strongly agree – strongly disagree), students found the learning objects generally somewhat effective in all measures (2.7), except when comparison with the textbook (3.4). Finally, as to usability, also on a scale of 1 to 5 (strongly agree – strongly disagree), the result was not very encouraging with a rating of 3.3, which indicates that the particular learning object design needed additional work.

In a similar case study, Costelloe, Sherry, and Magee (2009) determined what topics were the weakest for CS1 students in three higher-education institutions: arrays, repetitions, and selection, plus additional object-oriented topics in one of the three institutions. Based on the commonalities, the researchers opted to focus their attention on arrays and designed a comprehensive set of interactive exercises on the subtopics of array initialization, manipulation, and searching. These learning objects incorporated the scaffolding concept and provided various levels of visualization, animations, and automated feedback. Costelloe et al. (2009) evaluated this case study through an analysis of observer notes, student surveys, and the results of pre- and posttests, for both experimental and control groups. The t-test statistical analysis of the results indicated no statistical significance between the experimental and control groups. Some additional results were obtained from the observations of students using the learning objects, with many of the participants engaging well with the objects and issuing positive opinions about the experience.

Automated Tutors. Using automated tutors in CS1 courses has been reported as beneficial in the literature by Moritz, Wei, Parvez, and Blank (2005). The authors developed an automated intelligent tutoring system and implemented it with an experimental group and a control group, with random selection of participants, and a pretest/posttest of programming ability. The intelligent tutoring system used automated guidance based on each student's previous performance and offered feedback that depended on the type of exercise and the level of difficulty. The statistical analysis of the results indicated an improvement in the posttest for students exposed to the automated tutor: 72.5% with the automated tutor versus 62.3% without.

Soh (2006) implemented the Intelligent Learning Materials Delivery Agent (ILMDA) intelligent tutor in a CS1 course, which resulted in a positive experience. ILDMA used the

student's profile and performance to determine what topics, exercises, and assessments to present next. With the objective of determining the effectiveness and efficiency of the tutoring system, the researcher compared two groups of students: (1) using the Heuristic-ILMDA, exercises were presented based on encoded instructional strategies and (2) using the Learning-ILMDA plus the encoded instructional strategies, the system used machine learning algorithms to further help students in solving the more difficult exercises. The students in the Learning-ILMDA group reached the desired exercise difficulty level after reviewing an average of 1.7 examples and 6.7 problems; while the Heuristic-ILMDA group did it in an average of 2.4 examples and 11.1 problems. From these results, Soh concluded that the Learning-ILMDA tutoring system was more efficient for students. No control group was analyzed to compare the results.

Yoo, Yoo, Deo, Dong, and Pettey (2012), developed the AlgoTutor automated tutor for algorithm development, which they tested with three CS1 classes. The control group consisted of two classes which did not use AlgoTutor and one experimental group, which used the tutor. There was a significant improvement (9.5%) in algorithm-design knowledge for the experimental group (measured by successfully designing a set of algorithms). However, test results were not significantly different.

Student Teams. Since most software development is done by teams of developers, it is natural to teach programming through student teams, and several research studies proposed such an approach. For example, Gonzalez (2006) added active learning and cooperative learning techniques to a CS1 course and obtained a strong drop in DFW rates for CS2—that is, for the course following CS1. For CS1 itself, however, the addition of such techniques did not significantly improve the DFW rate, which is certainly unexpected and could merit additional research. The researcher noted that such an approach has been intensely studied in several

educational areas, particularly in science, technology, and math (STEM), so applying collaborative learning to CS1 seemed like a good idea, given the similarities between computer science and the other STEM disciplines. The face-to-face course had two 1.5-hour class sessions per week, each divided into three ordered sections: (1) group discussion of previous work, (2) a 30-minute lecture for the new material, and (3) a collaborative programming activity, putting the new concepts in practice. The study was developed over two full years and six different classes each, with an experimental group using the "active learning" component, and a control group. The grades earned in CS2 by the two CS1 groups differed in a small but significant percentage, 24.3%. As a non-definitive, anecdotal added experience, a small portion of students entering into CS2 were given an "entrance" test and in this case the students in the experimental group scored higher than the students in the control group, earning an average grade of 80% versus only 58.6% (a difference of 21.4 percentage points).

Williams, Wiebe, Yang, Ferzli, and Miller (2002) analyzed a collaborative learning experience called paired programming. Teams of two students were assigned to collaborate fully on their learning. The team was required to share one computer while designing and implementing the computer programs. Paired programming is an important tenet of "extreme programming" (Beck, 2000), which is a popular way to manage and organize software development projects. The experimental study was developed with two complete classes of CS1, both taught by the same instructor: the experimental group had collaborative programming labs and the control group had the standard, non-collaborative, solo programming labs. The student pairs were selected randomly by a computer program and the lab grades earned by each pair in the experimental group were heavily affected by the degree of collaboration achieved (measured by a peer evaluation). The results were encouraging. In the control group, the performance

(grades of "C" or above) was 45% but in the experimental group, the rate was 68%, an improvement of 23 percentage points. In both the midterms and final exam, there was also a marked difference with 63% in the control group and 73% in the experimental group. Finally, in the programming projects that were part of the assessment in the course, the control group earned 74% and the experimental group earned 85%, which represents an 11 percent increase.

Engaging Lab Exercises. A good way to engage students in CS1 is to include programming assignments that are attractive and that deal with subjects of interest to modern students. In the typical CS1 programming class, these examples tend to be "toy" programs that help demonstrate one or two programming concepts but that seldom perform any useful work. For example, Stevenson and Wagner (2006) determined that proper programming exercises should follow certain guidelines: be real-world based, allow realistic solutions to the problem, be interesting and challenging, and be focused on topics being discussed in class at that time. Some examples offered by the researchers include a Web crawler program, a spam evaluator, and an advanced string parser. Stevenson and Wagner did not report on the implementation of the guidelines for these exercises.

A different approach is to use a highly visual programming language such as Alice (Carnegie-Mellon, 2012; Cooper, Dann, & Pausch, 2000), instead of a traditional computer language, as analyzed by Garlick and Cankaya (2010). The researchers compared an experimental group using Alice in the first two weeks of the CS1 course (with 82 students) and a control group using pseudocode over the same period (with 72 students). The purpose was to determine if students liked the Alice version of the course better than the pseudocode version and how the performance compared between the two versions. The performance was somewhat discouraging, since the end-of-period exam grades for the Alice group were lower, although not

in a statistically significant way. In a survey offered to participants, the experimental group's mean responses were actually lower than the control group, which led the researchers to conclude that Alice needed to be analyzed further.

In a similar study, Price (2013) examined a CS1 course where Raptor (Carlisle, Wilson, Humphries, & Hadfield, 2004) was used for some topics (selection and repetition) and Alice (Cooper, Dann, & Pausch, 2000) for some other topics (objects and classes). Raptor is a programming environment that executes flowcharts directly, making for a highly visual programming environment aimed at general computational problems. Alice is also highly visual, although its objective is to implement graphical "scenarios" with animated images. The researchers developed a posttest, non-equivalent group, quasi-experimental design with the treatment group using both Raptor and Alice as programming tools, while the control group used standard, text-based programming languages. The results were encouraging: retention in the course was improved from 72% (control group) to 84% (treatment group), a positive difference of 12 percentage points. Performance, as measured by the final grade earned in the course, also had improvements: 68% to 72%, a positive but not significant difference of four percentage points.

Category 3: Live Instruction

Modeling. The process of programming—the translation of the algorithm into finished code—is a particularly difficult skill and most novice programmers struggle with it. To teach that process, researchers have proposed that the instructor should develop practical presentations of how to make that difficult transition from design to code. In the conclusion of an exhaustive review of research studies on teaching and learning programming, Robins, Rountree, and Rountree (2003) recommended explicitly teaching students the design strategies and methods to

design programs that experts programmers utilize, given that the salient characteristic of an expert versus a novice programmer is understanding algorithms and problem-solving strategies, and not computer language coding proficiency. Robins et al. (2003) reviewed computer programming, industry trends, characteristics of novice programmers, and instructional design considerations related to computer programming. In a comparison of expert and novice programmers, the researchers noted that experts tend to think about computing problem in terms of algorithms, patterns, and computational methods, while novice programmers tend to do the same based on less important details such as the actual computer code and its syntax. The essential difference, then, between experts and novices is the poor capacity for modeling that the novices have. This specific point indicates that an important component of an introduction to programming course should be an attempt to have students construct such mental models, thus the suggestion (Robins et al.) of having "live" lectures showing novices how an expert develops the program from problem analysis to the final code, clearly explaining what strategies are implemented.

Fjuk, Berge, Bennedsen, and Caspersen (2004) discussed the pedagogical foundation of such meetings, where online interactions between the instructor and students lead to learning based on the master-apprentice model. They concluded that this model holds great promise for the future, especially where the modeling sessions can be in real time. The case study was developed over an object-oriented CS1 hybrid course and the results were based on the researchers' observations during the course and interviews with a group of students, the instructor, and the teaching assistant. The researchers noted the importance of modeling the problem-solving strategies used by experts, in a master-apprentice relationship, so that novices can more effectively learn computer programming. With respect to the implementation of the

apprenticeship instructional strategy, the researchers introduced the concept of weekly assignments that have no bearing on the final grade of the course and that are how the modeling sessions between master and apprentice are implemented. These sessions were not developed one-on-one with each student but with the entire class and delivered via videoconferencing. Prior to this meeting, the students were required to contact the instructor with specific requests about the assignment. The intent was for students to work on the assignment before the modeling session so that the session would be more fruitful. These sessions were accompanied with brief videos posted in the classroom website where a single topic is explained and modeled.

When taking a CS1 course online, one of the techniques for improving its performance is through live, synchronous lectures via online conferencing systems such as Adobe Connect (Adobe, 2012). In a quasi-experimental study, Restrepo and Trefftz (2005) added synchronous lectures partially delivered through online conferencing software but with the added twist of simulated "telepresence" in the form of avatars in a virtual-reality graphical classroom. While the study was based on computer graphics and physics courses and not CS1, the concept is nevertheless intriguing due to the results reported. The treatment group received instruction using the synchronous lectures with the telepresence feature, while the control group received the instruction without it, in the traditional face-to-face teaching style. The results were obtained by grading the work produced by the two groups based on a scale of 1 through 4 (1 – naive; 2 – novice; 3 – apprentice; 4 – master). For the computer graphics course, the results favored the treatment group but the results were not statistically significant through a Mann-Whitney test, probably because of the small sample size of four students. For the physics course, however, the sample size was larger with ten students and results also favored the treatment group and were in addition statistically significant with a Student t-test of 0.047. These results led to the conclusion

that class lectures delivered through videoconferencing could effectively replace the traditional face-to-face lectures. No additional tests were done with and without the telepresence feature; therefore, that feature's effect on the results was not determined.

Category 4: Cognitive Apprenticeship

Several studies have been developed analyzing the implementation of a cognitive apprenticeship (CA) instructional strategy in a CS1 course. Mow, Wing, and Yates (2006), for example, showed that students significantly improve their learning with a CA-based approach, although this particular study was developed in a hybrid class format with the added component of collaborative learning, which is not a specific part of CA. The study was developed with a quasi-experimental design, with non-equivalent groups and only a posttest, plus an evaluative questionnaire to measure student satisfaction. The performance results indicated a significant improvement over the control group (without CA) by the treatment group, with 9 vs. 13.5 points, +50% (lower-performing students) and 15.5 vs. 17 points, +9.7% (best-performing students). The results of the questionnaire indicated similar results for both the control and treatment groups, with 34.1 out of a maximum of 44 score (the midpoint of the scale was 27.5), which indicates a good level of appeal for the course.

The implementation of these more heavily guided programming exercises is supported by results obtained by Hassinen and Mayra (2006), who found that extensive and intense practice is one good way to learn computer programming. This conclusion was reached after examining a CS1 course and also an advanced networks course taught over several semesters. Students were offered a small number of grade points for completing exercises, and the number of points earned increased based on the number of practice exercises that they completed correctly. Given

this grade incentive, the researchers determined that there is a strong positive correlation between the number of exercises completed by students and the final grade earned in the course.

Vihavainen, Paksula, Luukkainen, and Kurhila (2011) developed an extended CA model, which they named Extreme Apprenticeship (XA), and applied it to a face-to-face CS1 course, with significant improvements in performance. XA is based on CA plus an added and strong emphasis on developing guided programming exercises. As Vihavainen et al (2011) noted, a minimally-guided instructional strategy does not work well for complex cognitive learning such as computer programming (Kirschner & Sweller, 2006) and for this reason, they decided to expand CA in this way. Vihavainen et al. (2011) compared the final grades of students in non-XA courses offered to students over an eight-year period, with the final grades earned by students in one year of XA-modified courses. For the first part of the CS1 course (with basic topics) the long-term average for the non-XA courses was 51.1% and with XA the average increased to 70.7%, which is a positive difference of 19.1 percentage points. For the second part of the CS1 course (with more advanced topics) the results were also positive with the non-XA long-term average of 52.7% and XA average of 82%. In addition to these performance results, retention rates went from a 43.7% rate to 70.1%, an improvement of 26 percentage points. The XA concept adds to CA an accelerated feedback loop between master and apprentice, making feedback nearly immediate. In addition, the standard strategy is complemented by having numerous exercises as the basis for the modeling and scaffolding support and making the exercises an essential part of the assessment of the performance. The course was implemented with two-hour lectures per week and an online "textbook" that closely followed the exercises (instead of the other way around), so that the textbook itself served as part of the initial needed scaffolding to do the exercises successfully. The lectures were aimed mainly at modeling, so

that students could observe and interact with an expert while the solution to a given problem was being analyzed and implemented. In addition to the lectures and textbook, the main elements of the course were the exercises. First, basic skills were presented in a step-by-step fashion so that students could build knowledge. These exercises became more complex throughout the course. The exercises were developed in a physical computer lab environment, with instructors and teaching assistants present to provide support and feedback. One interesting feature of the exercises was a public checklist where each successful completion of an exercise was marked (without any grades – the assessments were private between instructor and each student). As such, students could compare their own progress with that of their classmates. The other components of the assessment were several biweekly exams and a final exam.

One year later, Vihavainen, Luukkainen and Kurhila (2012) applied their extreme XA model to a CS1 course taught as a massive open online course (MOOC) and had positive results. The authors used an automated tool called *Test My Code* to provide the scaffolding required by CA, so that it could also scale up to several hundreds of students simultaneously. The *Test My Code* tool was not described in detail but its most salient feature was a domain-specific language that allowed instructors to embed scaffolding comments and actions within the exercises, so that students could work on the exercises at their leisure and obtain useful feedback instantaneously. Even so, however, the course was supported by a staff of 20, including the instructors and a group of advanced, late-year, computer science students who acted as advisors for the students taking the course. The authors built all the course's materials, exams, and other features around the 173 exercises that comprise the course and many of the exercises build upon one another in order to form bigger, more complex programs. The automated exercise feedback appears to be a good option to implement the necessary scaffolding for students in a CA-style course. The study

was developed with the first offering of the MOOC and of the original 417 enrollees, only 70 (17%) finished the course. Large attrition is expected in MOOCs due to their nature (Thrun, 2012), so this dropout number is not surprising. In addition to the automated feedback provided by the Test My Code tool, the course also had an asynchronous forum for student-to-student support, which was also used frequently, particularly at night and on Sundays (due dates were set for Sunday evenings).

Application of CA to CS1

When offered in the face-to-face or blended learning classroom, a CS1 course taught using CA methods have offered good results, both in performance and retention (Vihavainen, Paksula, Luukkainen, & Kurhila, 2011). CA is derived from the classical style of learning in which many hand crafts are learned: a "master" in the craft teaches the "apprentice" how the craft is developed through modeling, starting from simple items and progressing towards more complex items, while all the time providing support and scaffolding to the apprentice. This scaffolding provided by the master is then gradually removed as the apprentice gains experience.

CA can be implemented in various ways but some of the essential elements are as follows (Collins, Brown, & Newman, 1989):

Modeling. The instructor offers periodic modeling sessions where the instructor demonstrates the thought processes developed while attempting to solve programming problems. The objective is not to demonstrate features of the computer language used but instead, to show students how the instructor would approach the solution to the problem.

Coaching. The instructor offers specific guidance to the student while working on exercises so that the correct approach is applied to the solution of the problems, with the objective of correcting performance deviations as soon as possible.

Scaffolding. Based on gradually more difficult exercises, the instructor leads the student toward ever more complex challenges, until the student reaches the needed learning objectives. As these exercises progress forward, the instructor's support is gradually removed, until at the end of the course practically no support is needed.

Articulation. Students must explain to the instructor why a specific approach to a problem works, so that the thinking process leading to the solution can be analyzed and solidified in the student's mind.

Reflection. After each main learning experience, a "reflection" paper is prepared where the students summarize what they have learned during the previous period and how it can be used in practice.

Exploration. Students are encouraged to try out new approaches to resolution of the problems, with the intent of developing independent thought.

Given that a significant number of students have difficulties when facing the CS1 course, and given the good results that CA has offered to CS1 courses when taught face-to-face or in hybrid classrooms, it appears reasonable to determine how well a CA approach will apply to a purely online CS1 course. This topic is interesting not only to the researcher, who has taught this course online for over ten years and continually seeks to improve retention and performance in it, but also for the computer science teaching community at large. The marked growth of online course offerings and the expected increase in computer science job offerings over the coming

years means that a CS1 course should continue to grow in popularity in the future, making the results of this study relevant as a novel approach to resolving the difficulties with CS1.

CA as an Instructional Strategy

Constructivism

The CA instructional strategy falls within the scope of social constructivism. Phillips and Soltis (2004) presented Jean Piaget and his seminal work on learning, which is currently known as "psychological constructivism." In psychological constructivism, humans are born with a few basic abilities, and through experience build ever more complex cognitive structures that serve to explain the environment, predict its behavior, and thus restore cognitive equilibrium. Children go through four stages in the development of these cognitive structures: (1) sensorimotor stage, from age zero to two, where the physical coordination of senses and movement are established. (2) Preoperational stage, from age two to seven, where the mind constructs networks of cognitive structures, relating them one to the other, although limited to concrete, physical concepts, not abstract. (3) Concrete operations stage, from age seven to 11, where the child takes the important step of beginning to think in abstract terms. And (4) formal operations, from age 11 to 14 or 15, where the full ability to solve abstract problems is attained, with cognitive structures approaching those of adults.

As a trained biologist, Piaget sought to explain learning in a biological manner, and he compared the human mind to a system that naturally seeks equilibrium. When a new event or concept is encountered, the mind temporarily loses its normal cognitive equilibrium and learning occurs by the strong need to restore that equilibrium. Thus, new or adapted cognitive structures are built that include such new events or concepts. It is precisely this concept of equilibrium that

casts the strongest shadows on Piaget's work, and which derives in other associated constructivist theories, such as social constructivism and "radical" constructivism.

Zone of Proximal Development

CA also incorporates Vygotsky's concept of zones of proximal development (Wass, Harland, & Mercer, 2011). Zone of proximal development refers to the social context of learning by describing the distance between the intensity of task difficulty that the student is capable of resolving at present as compared to the potential intensity of task difficulty that the student could learn to resolve as a result of the instructor's assistance. As such, the instructor and student work together within that zone of proximal development until the student can successfully go beyond that zone and define a wider, more complex zone of proximal development. This process of ever-widening zones of proximal development is repeated until the student fully masters the complete topic.

Scaffolding

An essential component of CA is scaffolding (Collins, Brown, & Newman, 1989). The process of providing instructor's assistance in just the level needed so that the student can successfully learn to expand the zone of proximal development to a higher level is called scaffolding. Pea (2004) explained that scaffolding is present in our learning from the very earliest stages of our lives when our mothers played peekaboo and many other such infant games with us. Scaffolding provides us with the needed assistance in performing a given task which we alone would be challenged to accomplish and this assistance must be of a fading or gradually diminishing nature, so that the learner eventually learns to accomplish the task independently of the instructor. This assistance can be of two types: channeling and modeling. Channeling refers

to the process of actively constraining the learner toward the specific actions that will lead toward a successful solution of the task, whereas modeling is the act of presenting the learner with a completely reasoned solution to a similar task, with emphasis on the thought process that leads to its solution.

Situated Cognition

Brown, Collins, and Duguid (1989) proposed that the context of learning is an important, not ancillary, consideration within the overall learning process. This notion is referred to as situated cognition. Learning is, in fact, a process of enculturation, where a given knowledge domain's culture –and not just its topics—are learned and acquired by students when they learn the authentic activities relevant to that domain. Authentic in this sense means those activities that are actually done by practitioners in the given knowledge domain when facing a problem. Situated cognition is an important characteristic of CA in the sense that the master-apprentice mentoring seeks to imbue the apprentice with the problem-solving approach used by the master when confronting a new problem.

Cognitive Load Theory

Learning complex topics such as computer programming requires intensive cognitive work. Cognitive load theory seeks to improve such learning by reducing unnecessary information in working memory. Von Merrienboer and Sweller (2005) described cognitive load theory by explaining that the human mind can only retain in working memory a handful of new elements before committing the new learning to long-term memory, and that it is only when such new facts are stored in long-term memory, associated with already-existing mental schemas, that learning occurs. Thus, the theory indicates that instructional design should reduce the

simultaneous elements presented at the same time to learners, so that the limited capacity of our short-term memory can process them and thus result in learning when stored in long-term memory.

Problem-Based Learning

Problem-based learning was developed at McMaster University in 1969 (Neville, 2008) and even though originally developed for medical training, it has been implemented as a learning strategy in many fields, including computer science education (Barg, Fekete, Greening, Hollands, Kay, & Kingston, 2000). The strategy is aimed at solving large, complex problems, not just simple, one-topic exercises and it is often structured around activities such as planning how the learning will take place, problem-definition brainstorming, in-depth explanation of the possible approaches to solve the problem, reflection on the outcomes of the learning, and peer interaction in the exploration and implementation of the solution to the problem. Greening (1998) identified scaffolding as an appropriate technique to improve the results of problem-based learning and Ge and Land (2004) argued that scaffolding –one of the basic tenets of CA—is helpful in resolving problems, particularly ill-defined problems. As such, the analysis of problem-based learning strategies offers important insights into scaffolding methodologies.

Summary

This review of the literature highlighted important topics and existing research pertaining to the proposed investigation of applying CA to the teaching of abstract and complex skills. The review had four main threads: (1) an analysis of the teaching of abstract and complex skills; (2) a review of the first computer science course, or CS1, including a brief review of solutions that have been tried to "solve" the CS1 problem; (3) a description of CA as an instructional strategy,

and the affordances of CA as an instructional strategy in the design of a fully online CS1 course; and (4) a review of the learning theories that support CA, including constructivism, zone of proximal development, scaffolding, situated cognition, cognitive load theory, and problem-based learning. The next chapter describes the design and development research methodology that guided the construction of the CS1 course and its subsequent internal validation.

Chapter Three

Methodology

Overview

The goal was to design and validate an online CS1 course that incorporates CA strategies. A design and development research approach (Richey & Klein, 2007) was used to guide the construction and internal validation of a fully online CS1 course. Richey and Klein defined design and development research as a way to "create knowledge grounded in data systematically derived from practice" (p. 1). The authors described the process of instructional design model construction and validation and explained the importance of using a systematic process to validate products as opposed to anecdotal evidence such as testimonials. In design and development research, theories are tested and ideas are generated to "establish new procedures, techniques, and tools based upon a methodical analysis of specific cases" (Richey & Klein, 2007, p. 1). For example, Tracey and Richey (2007) constructed and validated internally an instructional design model that incorporates the cognitive theory of multiple intelligences (Gardner, 1983) as an overlay. The authors used a systematic process to develop the model and then validated the model using the Delphi technique where a panel of experienced instructional designers analyzed and offered feedback on the proposed instructional design. Once the suggested revisions were analyzed and incorporated into the design after Delphi rounds one and two, by round 3 the model was agreed upon by the panel of experts. The validated instructional model resulting from the Delphi process was then validated externally in a subsequent study (Tracey, 2007), although external validation was beyond the scope of this study.

Richey (2005) described the difference between internal and external validation of instructional design models and proposed to support the design process by executing both internal and external validation of any new or modified model. The goal of internal validation is to check the integrity of the model, while external validation is used to evaluate the effects of using the model. Some of the questions that *internal* validation seeks to answer are the following: (1) Does the model have any steps missing / are all steps necessary? (2) Are the steps ordered in the appropriate sequence? (3) How does the model contribute to learning? (4) How well can the model be used in various learning situations and environments? (5) Can the steps in the model be implemented by both novice and expert instructors? (6) Are the implementation costs of the model reasonable? On the other hand, *external* validation seeks to answer some of the following questions: (1) How well does the model comply with the learning needs and requirements? (2) How interested, motivated, and engaged were learners while using the model's instructional steps? (3) How efficient and appealing is the learning? Richey suggested that expert review is a commonly used method for internal validation of an instructional design model and mentioned that the Delphi technique is a common method.

This expert-review process for model construction and validation was selected to construct and validate the online CS1 course. Following is a brief description of the process followed: First, an extensive review of the literature was conducted to identify instances where CA has been used to support the development of abstract and cognitive skills. Second, Morrison, Ross, Kalman, and Kemp's (2011) guidelines for the development of instructional objectives and related instructional strategies were used to guide the design of the CS1 course. Existing research literature pertaining to the application of CA in CS1 courses along with instructional design theory guided the selection and integration of CA strategies that were most appropriate for each

learning objective. Third, once the course was designed, it was validated using a Delphi technique (Dalkey & Helmer, 1963). The course guide was distributed to a group of seven instructors who teach (or have taught in the last three years) CS1 in an online or blended learning environment. Panel members were recruited by soliciting appropriate university alumni mailing lists (with permission). Panel members were provided with instructions to review the guide and offer suggestions for improvement. Fourth, data collected from the Delphi panel members was analyzed and used to modify the CA-supported course design.

Research Methods

There were two research phases: course construction and course validation. Following is a description of each phase followed by the specific procedures implemented.

Phase One: Course Construction

As part of the study's proposal, a preliminary course guide (Appendix B) was developed using Morrison, et al.'s (2011) guidelines for the development of instructional objectives and instructional strategies. They advised that prescriptions are needed for different types of content such as facts, concepts, principles, rules, and procedures. These prescriptions describe methods of instruction that are most appropriate for teaching the content in an efficient and effective manner. Morrison et al. suggested that in addition to the prescriptions for various types of content, the designer's experience also plays an important role in design of instruction. A guiding question that the authors offered is, "How do I present the content needed to achieve this objective?" The preliminary course guide provided detailed information to guide an instructor in the implementation of CS1 that incorporates CA strategies where appropriate. Instructional strategies included the learning objective, the content type or types (i.e., fact, concept, principle, rule, and procedure) and the expected performance level (i.e., recall or apply), the initial

presentation (i.e., how the information is presented to the learner), the generative strategy (i.e., activities that help the learner to process the information that was presented), and the assessment (i.e., how the learner will be assessed to determine whether he or she has mastered the objective) (Morrison, et al.). These elements were used to design and develop the preliminary course guide.

Phase Two: Course Validation

The second phase following course construction was internal validation of the course guide. Richey and Klein (2007) identified three different ways to conduct an internal validation including expert review, usability documentation, and component investigation. Expert review was selected as the validation method. Course validation was carried out using the Delphi technique, which concluded with consensus of the panel in round 3. The Delphi technique was conceived in the 1950s as part of a study developed for the US Department of Defense by the Rand Corporation (Dalkey & Helmer, 1963). The objective is to attain a reliable consensus of opinion from a panel of subject matter experts. The technique consists of iterative questioning of the panel of experts with regard to the subject at hand, with periodic feedback to the group regarding: (1) experts' requests for additional data that is deemed relevant to the evaluation and (2) additional or modified design elements suggested by one or more experts that could be relevant in reaching an opinion. This iterative process is developed in "rounds," where the opinions from a first round of questioning is analyzed, the requests for additional data and design suggestions are communicated back to the panel of experts for consideration, which results in a second round of questioning, and so on. This careful control of feedback to the panel of experts results in better, more pristine opinions, without the possible influence of strong personalities and without confrontations within the panel. As Dalkey and Helmer explain, the iterative round-to-round corrections replace each individual panel member's opinion by the consensual opinion

of all the experts. Mitroff and Turoff (1975) classifies the Delphi method within Lockean inquiry systems and explain that the validity of the result can be measured by how well consensus is reached among the panel of experts. This means that Delphi-derived decisions may not necessarily be the absolute "best" but be instead a compromise decision.

Even so, Dalkey and Helmer (1963) explained that it is unrealistic to expect complete consensus among the panel of experts, even after several rounds of analysis. To help reach better consensus, suitable corrections to the experts' replies for each round are made, taking into account the following parameters: (1) adjustments to basic assumptions, so as to reach a consensus; (2) perceived sensitivity of the panel members with respect to such changes in basic assumptions; and (3) perceived accuracy of the panel members in reaching an exact opinion.

Linstone and Turoff (1975) noted that one problem with the Delphi technique is that the researchers formulating the questions may influence the results by inserting a conscious or unconscious bias into the content and wording of the questions presented to the panel of experts. Even question wordiness may also have an important effect on the results, with either too few or too many words resulting in lower consensus. Word count of 20 – 25 words was determined to produce the best consensus in topics familiar to the panel; but for topics unfamiliar to the panel, longer word count increased consensus. Another important consideration is the effect on the panel of experts of round-to-round feedback and the opinion of the other members (Linstone & Turoff). Experiments using false feedback offered as a result that: (1) panel members are sensitive to the opinions of the other members and (2) there is a desire to achieve a consensus of opinion. Both results point toward the critical importance of round-to-round feedback on the consensus.

The Delphi technique has been applied to instructional design investigations as well. York and Ertmer (2011), for example, applied the Delphi technique to identify heuristics applicable to instructional design, with a panel of 31 experts. Of the 59 original heuristics (identified in a previous study by York), only 14 attained consensus by the panel after the three rounds and they were found to be relatively well-related to the International Board of Standards for Training, Performance and Instruction's instructional design competencies (IBSTPI, 2012). In another study, Neuman (1995) analyzed the use of document databases as references for high school students, with a 25-member panel of experts and a four-round Delphi method, which resulted in clear guidelines for the design of such databases and also for the design of the instruction that will use those databases, aimed at this particular group of students.

Procedures

Following are the specific procedures that were followed in the two phases: (1) initial course design and (2) internal validation of that course design through the Delphi technique.

Initial Course Design

The initial course design incorporated all elements of CA in a format suitable to an online asynchronous classroom as implemented through a learning management system such as eCollege (Pearson, 2014). Following is a description of the CA elements that were included in the CS1 course design:

1. *Modeling*. The instructor offers periodic modeling sessions where the instructor demonstrates the thought processes developed while attempting to solve programming problems. The objective is not to demonstrate features of the computer language used but instead, to show students how the instructor would approach the solution to the problem.

eCollege Implementation. Instructor-led videoconference session.

2. *Coaching*. The instructor offers specific guidance to the student while working on exercises so that the correct approach is applied to the solution of the problems, with the objective of correcting performance deviations as soon as possible.
   eCollege Implementation. Individual asynchronous exercise sessions.

3. *Scaffolding*. Based on gradually more difficult exercises, the instructor leads the student toward ever more complex challenges, until the student reaches the needed learning objectives. As these exercises progress forward, the instructor's support is gradually removed, until at the end of the course practically no support is needed.
   eCollege Implementation. Individual asynchronous exercise sessions.

4. *Articulation*. Students must explain to the instructor why a specific approach to a problem works, so that the thinking process leading to the solution can be analyzed and solidified in the student's mind.
   eCollege Implementation. Individual Forum "Reflection" essay.

5. *Reflection*. After each main learning experience, a "reflection" exercise is developed where the student summarizes what they have learned during the previous period and how it can be used in practice.
6. *eCollege Implementation*. Individual Forum "Reflection" essay.

7. *Exploration*. Students are encouraged to try out new approaches to resolution of the problems, with the intent of developing independent thought.
   eCollege Implementation. Discussion forum participation and individual forum exercise discussion.

Delphi Validation

Prior to soliciting panel members for participation, permission was sought and obtained from Nova Southeastern University's Institutional Review Board (IRB) (Appendix K). Once the permission was obtained, an inquiry was sent to 169 Nova Southeastern University alumni with Masters or PhD degrees in computing-related areas, to determine: (1) if they teach or recently taught a CS1 course in the online or hybrid modality; and (2) if they would be interested in participating in the Delphi panel. The response from the inquiry was from 45 alumni (26.6%) and of those 17 (37.7% of replies; 10% of total) replied that yes, they would be interested in participating in the panel. After a detailed analysis of the affirmative replies, eight panel members were chosen, with six of the eight having had the required experience teaching CS1 and two additional panel members that had not taught CS1 but who did teach online computing-related courses, such as introductory computer courses (the so-called "CS0" course) for one and the other teaching more advanced courses, such as networking. The academic credentials of all eight panel members included a PhD degree.

Of the eight panel members chosen, one declined participation in round 3 due to ill health. Although the opinions of this panel member were included in the analysis for rounds one and two, it was not possible to include the panel member's opinion in the last round. Once the panel of experts was selected and confirmed as willing to participate in the study, an email (Appendix A) was sent to panel participants that included an overview of the study, instructions for round 1, the preliminary course guide (Appendix B), and a summary document of CA strategies (Appendix C). The experts were requested to reply within two weeks. The results were categorized as follows: (1) effectiveness, (2) efficiency, (3) appeal, and (4) CA. The categorized replies were used to modify the course guide and prepare the questions for the second round.

These questions were similar to the ones in the first round, except for some minor editing to make them clearer and the addition of a Likert scale to better gauge the level of agreement with each question. A fourth round was not needed since the panel reached consensus on the third round by replying in the affirmative that the course design: (1) appropriately incorporates the CA strategies; and (2) is sufficient for the delivery of the course in an online environment. Figure 1 is an illustration of the data collection and analysis process for phase two.

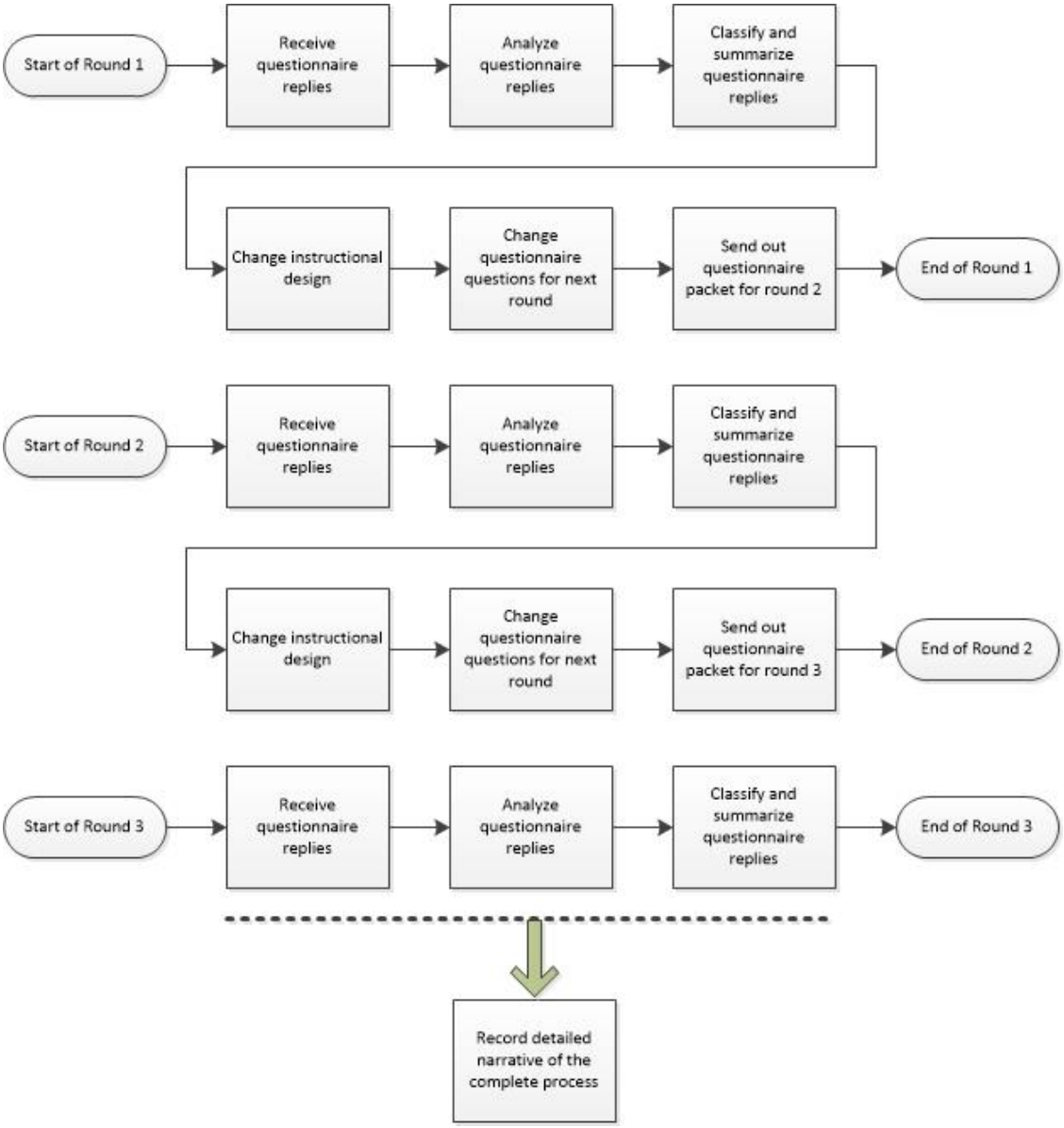

Figure 1. Phase two data collection and analysis process

Chapter Four

Results

The goal was to design and validate internally an online CS1 course that incorporates CA strategies. Chapter 3 presented the two-phase design and development research approach that was used to guide the construction and internal validation of a fully online CS1 course. Phase one resulted in the design and development of the course guide. An expert-review process (Richey & Klein, 2007) using the Delphi technique (Dalkey & Helmer, 1963) was implemented in phase two to validate the design with regard to its effectiveness, efficiency, and appeal (Reigeluth & Frick, 1999). Three iterations of the course guide were developed resulting in consensus by the panel and a final iteration of the guide in round 3. The following chapter presents results from phase two with a round-by-round analysis of the Delphi process and the results obtained following the panel's consensus.

Delphi Panel Round 1

In round 1, a set of documents was sent by email to each of the eight panel members. The packet included: (1) an explanatory email containing the instructions and questions (Appendix A); (2) the preliminary course guide (Appendix B); and (3) a brief description of CA including references so that panel members could find more information about CA, if desired (Appendix C). Panel members responded to the following questions. No specific format for replying was requested. Round 1 questions reflected the research questions as shown in Table 1.

1. *Effectiveness*: If you were to implement this course as designed, do you think it would be effective in helping students gain the knowledge and skills needed to achieve the stated learning outcomes? Yes/No. If no, please explain.
2. *Efficiency*: If you were to implement this course as designed, do you think you could deliver it in an efficient manner? Yes/No. If no, please explain.

3. *Appeal*: If you were to implement this course as designed, do you think it would be appealing to students? Yes/No. If no, please explain.
4. *Cognitive apprenticeship*: Do you think the cognitive apprenticeship strategies as described within the course guide are appropriate? That is, do the strategies support the type of content that is being taught? Yes/No. If no, please explain.
5. *Additional thoughts*: In addition to your comments above, what specific comments, questions, and/or recommendations do you have for improving the effectiveness, efficiency, and appeal of this course design? What suggestions do you have for improving the clarity of the course guide in general? Please refer to sections and or page numbers where appropriate.

Table 1. Correspondence of Research Questions with Round 1 Questions

Research Question	**Round 1 Survey Question**
RQ1. What is the evidence that specific elements of CA are helpful to students in an online CS1 and similar online courses that require abstract and complex thinking?	N/A (The answer to this question resulted in a review of the literature that guided the design and development of the preliminary course guide in phase 1.)
RQ2. How must the traditional course be revised to incorporate the selected CA elements for effective online delivery?	Question 4.
RQ3. What are the reactions of experienced instructors of CS1 to the proposed CA-supported online course in terms of effectiveness, efficiency, and appeal?	Questions 1, 2, and 3.
RQ4: What are the modifications needed to improve the proposed CA-supported CS1 online course in terms of perceived effectiveness, efficiency, and appeal?	Questions 1, 2, 3, and 5.

Once the comments from all of the panel members were received, the data were organized into the following categories: (1) efficiency concerns; (2) course design concerns; and (3) need for clarifications. The efficiency concerns were centered on the relatively large number of academic activities that the instructor must develop. For example, one panel member commented:

> *Depending on the number of students, personalizing exercises for each student may take a significant amount of time…and giving quick feedback by video could be a challenge for some instructors. If they find themselves "re-recording" the feedback, it would easily take 30 minutes or more to give feedback. While these types of interactions are admirable and very helpful to students, they may not be as feasible for larger courses. The instructor may need to limit detailed personal feedback.*

Another panel member was concerned about the amount of time that the students would have to spend on the course as designed and wrote:

> *The overall flow of the 8-week course requires students to read chapters, write essays, participate in discussion boards, watch videos, and develop programs. I realize that an 8-week course requires information to flow more quickly than other courses, but it seems to be quite an undertaking for the average student.*

To address these concerns, and after careful consideration of all CA-related elements of the course design, the weekly quizzes were removed. Removal of the quizzes reduces the amount of time spent on the course for instructors and students while at the same time places more emphasis in the hands-on elements of the design, such as programming problems and exercises. The suggested points for each activity were also redistributed to reflect this modification.

The second category of concerns referred to course design issues. One panel member noted:

> *I understand the reasoning for the reflective essay. This would ensure that the online students are completing their own work instead of finding solutions elsewhere. Another way that this could be approached is have the student write a project plan (pseudocode) and algorithm before coding the problem. Listing the methods that they plan on using to*

solve the problem. Another way that they could explore and/or reflect would be to give the student another student's solution and determine what their algorithm was and what methods they used to program the solution. If you didn't want to share the solutions of other students then you could give them a solution that you have come up with and ask them the same questions. The exchanging of programs will also show the student that there may be more than one way to solve a problem and maybe contrast which program might work better and not just work. This could also be used as a discussion forum that might be more interesting than rehashing the concepts from the lesson.

After reviewing these suggestions and analyzing the essence of the articulation component of CA, which the reflection essay aimed to implement, the reflection essay was incorporated into a weekly "Learning Reflection / Solution Analysis Essay" where students analyze the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used by the instructor and why. The students also compare their own coding techniques used in their program with that standard solution and explain the differences.

The third category of issues detected by the panel referred to the need for clarifications in the course design. The clarifications suggested (and implemented in the modified course design for Round 2) included:

1. Expand the reference to the course from "CS1" to "Introduction to Programming / CS1".

2. Clarify that the five-minute feedback videos are accompanied by a written rubric where the major points of the program are analyzed and graded. The rubric is done first, and then the video.
3. Add grading rubrics for the programming problems.
4. Clarify that this course is typically preceded by a so-called “CS0” course, where the elements of problem-solving with a computer program are introduced.
5. Clarify that the complete library of short instructional videos is available from day one until the last day of class.
6. Clarify that the programming software (Microsoft Visual Studio Express Edition) is a free download from the Microsoft website, and that it allows full use of the Interactive Development Environment and all basic programming features needed for the course. The textbook is also available online through CourseSmart at a reduced cost when compared with the printed version. The textbook also has a “student companion website” where additional videos and slide shows are included, free of charge.
7. Clarify that the individual discussion forums are separate from the classroom discussion forums.
8. Clarify that the last week's programming problem incorporates material from all previous weeks, including the Week 7's materials and as such it can be considered a final project, even if limited in scope. Such an approach, together with the comprehensive final exam, allows making an overall assessment of the student's learning.

Delphi Panel Round 2

For the second round, a packet of documents was emailed (Appendix D) to the panel of experts, consisting of a document of revisions noting all the changes made to the round-one course guide (Appendix D), and two versions of the new course guide: (1) original, round-one course design guide, with all changes and additions marked-up and highlighted in yellow, for easy identification; and (2) the complete round-two course guide, with all changes and additions incorporated in normal text (Appendix F).

The questions for the panel members were slightly modified to remind the panel members how the terms effectiveness, efficiency, and appeal were operationally defined and a Likert scale was added to better gauge the degree of acceptance of each question, ranging from 1 to 5 (1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree). The correspondence between the research questions and the round 2 questions is shown in Table 2. Following are the Round 2 questions:

1. *Effectiveness*: The course design is effective (i.e., the learning activities align with the stated goals and objectives of the course): 1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree. Comments/Suggestions for Improvement: _____.
2. *Efficiency*: The course design is efficient (i.e., the goals and objectives of the course are achieved using the least amount of resources possible): 1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree. Comments/Suggestions for Improvement: _____.
3. *Appeal*: The course design is appealing (i.e., the learning activities are designed to engage students in the learning experience): 1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree. Comments/Suggestions for Improvement: _____.

4. *Cognitive apprenticeship*: The cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught): 1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree. Comments/Suggestions for Improvement: _____.
5. *Additional thoughts*: In addition to your comments above, what specific comments, questions, and/or recommendations do you have for improving the effectiveness, efficiency, and appeal of this modified course design? What suggestions do you have for improving the clarity of the course guide in general? Please refer to sections and or page numbers where appropriate.

Table 2. Correspondence of Research Questions with Round 2 Questions

Research Question	**Round 2 Survey Question**
RQ1. What is the evidence that specific elements of CA are helpful to students in an online CS1 and similar online courses that require abstract and complex thinking?	N/A (The answer to this question resulted in a review of the literature that guided the design and development the preliminary course guide in phase 1.)
RQ2. How must the traditional course be revised to incorporate the selected CA elements for effective online delivery?	Questions 4 and 5.
RQ3. What are the reactions of experienced instructors of CS1 to the proposed CA-supported online course in terms of effectiveness, efficiency, and appeal?	Question 1, 2, and 3.
RQ4: What are the modifications needed to improve the proposed CA-supported CS1 online course in terms of perceived effectiveness, efficiency, and appeal?	Questions 1, 2, 3, 4 and 5.

The panel offered positive replies of "agree" or "strongly agree" for effectiveness (question one) and appeal (question three). For question four (cognitive apprenticeship), six of the seven panel members also replied with "agree" or "strongly agree," except for one panel member, who did confirm that the course design properly addresses CA requirements but replied as "undecided" since "I am not sure it would happen for me if I attempted to teach this course online." Upon further investigation of this reply, the concern had to do with the panel member's

ability to implement all the CA-related features instead of with the CA implementation itself -- in other words, with efficiency, not with CA itself.

With regard to question two (pertaining to efficiency) there were some concerns from one panel member about the ability of instructors to handle the academic load imposed by the various CA features. Specifically, this panel member noted:

> *Overall I think the course will work well, but am still hesitant that the instructor will not be able [to] "keep up" with timely feedback for students. I think your dropping the quizzes is helpful in reducing the number of assignments that students need to complete and that the instructor needs to grade.*

The final category of issues detected by the panel refers to the need for clarifications and minor modifications in the course design. The clarifications and modifications suggested (and implemented) included the following: (1) monitor student progress and quickly identify "at -risk" students, so that some remedial action can be implemented; (2) add a follow-up to the previous week's programming problem feedback, interacting with the student to clarify / help with doubts or issues; (3) reduce grading weight for class discussion and also suggest lesser participation than normal for other classes (set in the institution's policy); (4) expand rubrics so they show the level of work that will earn 100% / 70% / 30% / 0% of the grade for each rubric item; (5) clarify that the important topic of testing and debugging is included.

Delphi Panel Round 3

For the third round, an email (Appendix G) was sent to the panel of experts with attachments including a document of revisions noting all the changes made to the round-two course guide (Appendix H), and two versions of the new course guide: (1) original, round-two course design guide, with all changes and additions marked-up and highlighted in yellow, for

easy identification; and (2) the complete round-three course guide, with all changes and additions incorporated in normal text (Appendix I). It was not possible to obtain feedback from one of the eight panel members within the allotted time due to ill health and this panel member was dropped from the panel. Therefore, the third round only had seven of the eight original panel members.

Given the positive results from round 2, consensus was sought during round 3 by modifying the questions posed to the panel. As such, two questions required a yes / no answer and were aimed at determining if: (1) the cognitive apprenticeship strategies as described within the updated course guide supported the course content; and (2) the proposed guide was sufficient for the delivery of the course in an online environment. Round 3 questions included the following:

1. Do you agree that the cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught)?
2. Do you agree that the proposed guide is sufficient for the delivery of the course in an online environment?
3. If you disagree with 1 and/or 2, please explain what additional revisions to the course guide are necessary in order to meet these two objectives.
4. Please provide any additional comments or questions you might have.

The panel members were in unanimous agreement on questions one and two, that is: (1) The cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught) and (2) The proposed guide is sufficient for the delivery of the course in an online environment. Question three did not receive any replies (given the unanimous agreement) but in question four there were additional comments that merited consideration. One comment referred to “at-risk” students. The panel

member asked: *"What is your criteria for determining whether or not a student is 'at risk'? Grades? If so, is there a timeline for making this determination?"* The round 3 course guide contained the following remark in the section "Suggested Class Policies":

> *"At-risk" students should be identified early in the class and offered help. The nature of the topics in this course, which build upon one another, imply that failure in one topic will cause problems for the student in the next topic, so early detection and help are recommended. The instructor should keep close tabs on how well students are doing in the course, particularly with the programming problems, and take action as soon as poor performance is detected.*

Another comment referred to the possible issue resulting from large class sizes and the increased academic load that implementing the course design requires. The panel member suggested:

> *The asynchronous feedback discussion for programming assignments will be helpful to students. It will be important for instructors to give good feedback. This could also be a challenge for some instructors, especially if there are a large number of students in the course. The best test will be to run the course several times to see how students perform and how well instructors can implement the cognitive apprenticeship strategies.*

Course implementation is discussed in Chapter 5 in recommendations for future research. Once all panel members' replies were analyzed and it was determined that there was unanimous agreement on questions 2 and 3, a final email message was sent to the panel confirming this result and thanking them for their participation in the study (Appendix J).

Summary

A course guide for the online delivery of a CS1 course using CA strategies was developed. A Delphi panel of eight instructors of online or blended CS1 (or similar) courses was used to conduct an internal validation of the course guide and answer research questions two through four. Three iterations of the course guide were developed resulting in a final iteration in round 3. Recommendations from the panel members during each round were incorporated into subsequent iterations of the guide. The validated CS1 guide serves as a starting point for external validation. The external validation process is discussed in more detail in Chapter 5.

Given that a significant number of students have difficulties when facing the CS1 course, and given the good results that CA has offered to CS1 courses when taught face-to-face or in a blended classroom, it is reasonable to determine how well a CA approach will apply to a fully online CS1 course. Validating the instructional design of an online CS1 course that incorporates CA strategies helps the computer science teaching community at large. The marked growth of online course offerings and the expected increase in computer science job offerings over the coming years means that online CS1 courses should continue to grow in popularity in the future, making the results of this study relevant as a novel approach to resolving the difficulties with learning abstract and complex skills, such as those presented in CS1, when taught online. The following chapter presents conclusions, implications, and recommendations for future research.

Chapter Five

Conclusions, Implications, Recommendations, and Summary

Phase one resulted in the design and development of a CS1 course guide that incorporates CA strategies. This guide was designed for implementation of CS1 in a fully online learning environment. An expert-review process (Richey & Klein, 2007) using the Delphi technique (Dalkey & Helmer, 1963) was implemented in phase two to validate the design with regard to its effectiveness, efficiency, and appeal. Three iterations of the course guide were developed resulting in consensus by the panel and a final iteration of the guide in round 3. Results confirmed the application of CA as an effective, efficient, and appealing instructional strategy to use when designing an online CS1 course. In the following chapter, conclusions are drawn based on the research questions posed. Implications of the conclusions within the field of computing technology in education are presented and recommendations for future research are offered.

Conclusions

Research Question 1

A literature review of relevant research pertaining to this investigation was conducted to answer research question 1, *What is the evidence that specific elements of CA are helpful to students in an online CS1 and similar online courses that require abstract and complex thinking?* As part of phase 1, four main research areas were reviewed including: 1) an analysis of teaching of abstract and complex skills; 2) a review of the first computer science course (CS1), including a review of solutions that have been tried to solve the CS1 problem; 3) a description of CA as an instructional strategy and the affordances of CA as an instructional strategy in the design of a

fully online CS1 course; and 4) a review of the learning theories that support CA, including constructivism, zone of proximal development, scaffolding, situated cognition, cognitive load theory, and problem-based learning. The evidence revealed that CA, when applied to courses requiring abstract and complex thinking such as CS1, produces positive results particularly in student engagement and retention. From the earliest days, CA has been applied to the learning of complex topics such as mathematics (Collins, et al., 1989). For example, Bouta and Paraskeva (2013) applied CA to the teaching of mathematics in a graphical 3D environment and obtained good results in student engagement. Keijonen, Kurhila, and Vihavainen (2013) implemented a derivative of CA, which they call "extreme apprenticeship" in CS1 courses over three years and the results shown an improvement in retention of 25.6% (an average of the three years prior to the implementation compared to the most recent three years of the same course with CA).

Research Question 2

A preliminary course guide was developed using Morrison, et al.'s (2011) guidelines for the development of instructional objectives and instructional strategies to answer research question 2, *How must the traditional course be revised to incorporate the selected CA elements for effective online delivery?* Also as part of phase 1, the researcher used the review of the literature pertaining to effective implementation of CA in courses similar to CS1, along with Morrison et al.'s guidelines for the development of instructional objectives and related instructional strategies to guide the design of the CS1 course. Each instructional strategy included the learning objective, the content type or types (i.e., fact, concept, principle, rule, and procedure), the expected performance level (i.e., recall or apply), the initial presentation (i.e., how the information is presented to the learner), the generative strategy (i.e., activities that help the learner to process the information that was presented), and the assessment (i.e., how the

learner will be assessed to determine whether he or she has mastered the objective) (Morrison, et al.). CA strategies that were appropriate to the objective and content type(s) were selected to support the presentation and practice of the content.

Research Questions 3 and 4

Research question 3, *What are the reactions of experienced instructors of CS1 to the proposed CA-supported online course in terms of effectiveness, efficiency, and appeal?* and research question 4, *What are the modifications needed to improve the proposed CA-supported CS1 online course in terms of perceived effectiveness, efficiency, and appeal?*, were answered in phase 2. The panel reviewed the preliminary course design (Appendix B) and revealed some improvements that could be made with the criterion of efficiency (i.e., achieving the proposed goals of the course with the least amount of resources possible). They commented that some of the instructional strategies placed too much work on the on the instructor including: (1) leading the individual discussion forums with each student with many exercises to be evaluated and discussed; (2) preparing video feedback to each student on the programming assignment for each week; (3) evaluating the weekly reflection essays; (4) holding the interactive modeling sessions; (5) preparing the library of short, one-topic videos; (6) performing the "other" usual academic activities in an online course, such as writing lectures, leading and grading public discussion forums, grading quizzes and exams, and grading programming problems. In the resulting course design (Appendix I), the weekly quizzes have been removed and the individual discussion forums were combined with a reflection essay. These changes maintained the CA philosophy intact but improved efficiency in the consensual opinion of the Delphi panel.

In addition to course modifications that were made to improve efficiency, research questions 3 and 4 also gave rise to recommendations made by the panel that related to appeal

(i.e., learning activities designed to engage students in the learning experience). One of these design changes involved the reflection essay. Some panel members opined that in a computer programming course an essay would seem somewhat out of place and the course would thus lose some appeal. To address this concern, the reflection essay was modified to an analysis that the student must make of the previous week's standard programming problem solution and a comparison with the student's own solution. This modification improves appeal while at the same time conserves the CA-related reflection on the work done and how it was approached. Another modification incorporated into the final course design involved adding detailed programming problem rubrics, which are shared with students beforehand and point out what specific assessment items will be evaluated by the instructor.

Suggestions to modify the course design pertained to the criteria of efficiency and appeal. There were no specific recommendations for improving effectiveness. Panel members agreed that the proposed course design aligned with the stated goals and objectives of the course. In summary, the changes incorporated into the final course design included the following:

1. Elimination of weekly quizzes (efficiency).
2. Merge of the individual discussion forums with the reflection essay (efficiency).
3. Modification of the reflection essay to be a computer programming problem discussion / reflection (appeal).
4. Addition of detailed rubrics (appeal).

Implications

The CS1 course is the undergraduate student's first encounter with computer programming. The course is notoriously difficult and results in high drop-out rates. Caspersen

(2007) reported a world-wide average drop-out rate of 33% for all types of delivery modalities (face-to-face, hybrid, and online). When CS1 is taught online, the drop-out rate is even higher, closer to 50% (Vilner, Zur, & Sagi, 2012). To compound the problem, Barnes, Richter, Powell, Chaffin, and Godwin (2007) explained that the enrollment and graduation rates in computer science are declining even though the demand for computing jobs is rising. This trend equates to a scarcity of new computer science graduates precisely at a time when they are most needed by the job market.

Together with this problem, online course enrollments are at an all-time high, with 33.5% of current students taking at least one online course (Allen & Seaman, 2014). Therefore, it becomes important to determine ways to improve the outcomes of a CS1 course when taught online. Given the success of CA in teaching complex and abstract topics such as computer programming in a traditional face-to-face setting and even in some blended learning environments, it is useful for practicing instructors of CS1 to have a CS1 course guide that incorporates the CA strategies. This course guide (Appendix J) could also be used as a valid basis for designing CS1 courses.

Stepping beyond the bounds of this study, which dealt with applying CA to teaching abstract and complex skills in an online learning environment, the application of CA as described here could have important implications in online learning in general. Higher education has been powerfully impacted by massive open online courses (MOOCs) in recent years (Dasarathy, Sullivan, Schmidt, Fisher, & Porter, 2014) and it appears that not only these online courses will continue to grow in popularity and appeal but also that the MOOC technologies and philosophy will affect many, if not all, areas of higher education in the near future. Where does that leave the large number of universities teaching traditional online courses? The proliferation of the

low-price or even free high-quality online higher education that MOOCs promise to provide does not necessarily mean that non-MOOC, "traditional" online education disappears but it will evolve. While MOOCs could serve well large swaths of highly-motivated students (Dasarathy et al, 2014), there are still large contingents of online students that could be better served with a higher degree of contact between faculty and student. Online courses designed with CA learning strategies, in computer science or any other discipline, require such higher contact between faculty and students, particularly with the specific implementation of CA features that the course design (Appendix J) incorporates. As such, the CA-inspired online course could best serve this important group of students, even in the age of MOOCs.

Recommendations

The internal validation process that resulted from the expert-review and Delphi technique offers intriguing insights into course design and proved to be a very good way to discover alternative and more efficient ways to implement various elements of CA, such as scaffolding and reflection. Use of the Delphi technique in similar studies that aim to validate instructional designs in online environments and other emerging delivery systems could serve as a useful strategy.

Most of the panel members' suggestions for change were related to attaining a more efficient course design. Based on this finding, efficiency is an aspect of instructional design that has high importance for the panel members. Gannon, Ley, Crawford, and Warner (2009), for example, note that "concern about faculty workload" is an important inhibitor to participating in distance education. In addition, Reid (2014) notes that instructors are not properly incentivized for the adoption of new technologies, compounding the situation further, and then goes on to conclude that the new and necessary business orientation that higher education has necessarily taken in recent years only serves to place an enhanced emphasis in improved efficiency of the teaching process.

In view of such concerns about efficiency, the important scaffolding component of CA could be implemented in a different way through commercially available automated exercise software such as CodeLab (Yuliya, Zingaro & Petersen, 2014) and Pex4Fun (Tillmann, de Halleux, Xie, Gulwani, & Bishop, 2013). Test My Code (Vihavainen, Vikberg, Luukkainen, & Partel, 2013) is a similar automated exercise software but it is open source. Such an implementation could make an important contribution not only to efficiency but also to appeal, since students would thus be able to obtain acceptable feedback on their programming exercises

in just a few seconds. In fact, Tillman et al. (2014) report that using such an automated tool (Pex4Fun in this case) resulted in a significant reduction of faculty hours in grading programming problems and, at the same time, resulted in increased anecdotal appeal from students, who like the immediate evaluative response of the tool.

An important next step in this line of research would be to externally validate the course design; that is, to evaluate the effects of actually using the course design in an online CS1 classroom. As noted by Richey (2005) some of the objectives of that external validation would be to determine: (1) how well the course design "fit" the learning needs of a particular student population; (2) how much appeal the course design has for these students; and (3) how efficient the implementation will be. Morrison et al. (2011) stated the common misconception that what instructional designers develop is "worthwhile and successful" (p. 354). However, in order to truly confirm this value, a summative evaluation is recommended. Issues pertaining to measurement of learning outcomes; cost related to development and implementation, as well as, continuing costs; and attitudes or reactions to the course can be objectively examined through a summative evaluation.

Summary

In 1978, the curriculum committee of the Association for Computing Machinery (ACM) published recommendations for undergraduate computer science programs (Austing, et al., 1979). Computer Science 1 (CS1) is the first undergraduate course that computer science majors are required to take. Similar to subjects like math and science, computer programming can be a difficult subject because of the abstract and complex skills that are involved. This cognitive complexity can be compounded when CS1 is taught in an online classroom because of the need

to provide hands-on guidance and scaffolding. Cognitive apprenticeship (CA) (Collins, et al., 1989) is a constructivist instructional strategy that has proven successful in teaching courses in abstract and complex topics including mathematics, science, and computer programming in traditional face-to-face and blended learning environments. However, limited studies are available that report the application of CA in a fully online computer programming course.

The goal was to design and validate internally an online CS1 course guide that incorporates CS1 strategies. Internal validation is the "empirical verification of the components and processes included in a design and development model" (p. 158). Here, internal validation refers to the validation of the application of CA strategies to teach the facts, concepts, principles, rules, and procedures that comprise a CS1 course delivered fully online. When considering the validation of instructional design theories, models, and methods it is important to consider the criterion of effectiveness efficiency, and appeal. Reigeluth and Frick (1999) warn that these criteria are sensitive to the context of the instruction so the needs and wants of the stakeholders should be considered. Therefore, using a design and development research method (Richey & Klein, 2007) the following research questions guided this investigation:

RQ1. What is the evidence that specific elements of CA are helpful to students in an online CS1 and similar online courses that require abstract and complex thinking?

RQ2. How must the traditional course be revised to incorporate the selected CA elements for effective online delivery?

RQ3. What are the reactions of experienced instructors of CS1 to the proposed CA-supported online course in terms of effectiveness, efficiency, and appeal?

RQ4: What are the modifications needed to improve the proposed CA-supported CS1 online course in terms of perceived effectiveness, efficiency, and appeal?

Phase 1 focused on the preliminary design and development of a course guide. The researcher applied his own personal experience teaching CS1 online along with evidence from the research literature of successful CA strategies to develop the first draft of the guide. Using Morrison et al.'s (2011) guidelines on the development of instructional strategies, objectives and detailed instructional strategies were developed. This guide provided detailed information to guide an instructor in the implementation of CS1 that incorporates CA strategies where appropriate.

In phase 2, following the course construction, the expert-review technique (Richey & Klein, 2007) was selected to validate the course guide. Specifically, a Delphi technique (Dalkey & Helmer, 1963) was implemented. The technique consists of iterative questioning of a panel of experts with periodic feedback from the panel. Seven panel members participated. These panel members were selected out of a group 17 who responded to a solicitation for participation. These seven panel members were selected based on their experience teaching CS1 or a similar course in an online or blended learning environment. In round 1, a package of information including the course guide, details about CA, and a set of specific questions relating to the research questions were sent to the panel for their review and input. Input from the first round was analyzed and the course guide was modified accordingly. There were two subsequent rounds with a similar review, analysis, and revision cycle with consensus being reached by the panel in the third round.

Results from this design and development effort include a course guide for teaching CS1 using CA strategies that can be used by instructors who teach CS1 in a fully online learning environment. This guide has been reviewed and validated by a panel of experts for its effectiveness (i.e., the course guide aligns with the stated goals and objectives of the course), efficiency (i.e., the course guide achieves the proposed goals of the course with the least amount

of resources possible), and appeal (i.e., learning activities are designed to engage students in the learning experience). A logical next step following internal validation is external validation. External validation aims to "confirm the model not by verifying its components, but by documenting the impact of the model's use" (p. 12). As a suggestion for future research, it is recommended that this course guide be taken and implemented to evaluate its effectiveness, efficiency, and appeal among learners, instructors, and CS programs. As a suggestion for practitioners, it is hoped that the course guide can be used to help practitioners implement a fully online CS1 course that uses CA strategies to achieve the learning outcomes of a CS1 course while at the same time engage students in the development of abstract and complex skills that are inherent in computer programming courses.

Appendix A

Round 1 -- E-mail to Panel of Experts

Dear Dr. Xxx:

Thank you again for agreeing to participate in my dissertation research study, *A Cognitive Apprenticeship Approach for Teaching Abstract and Complex Skills in an Online Learning Environment*. The purpose is to construct and validate an online CS1 course that incorporates cognitive apprenticeship (CA) strategies that aid in teaching abstract and complex skills, such as those found in CS1 courses. Given your expertise in teaching this course or similar, your role is to help me improve a CS1 course guide that I developed that incorporates the use of CA strategies in a fully online learning environment. You will participate in a series of three rounds of reviews. The first round will likely take the most amount of time (approximately 1-3 hours) with subsequent rounds taking less time (approx. 20-60 minutes).

Following are instructions for Round 1. These instructions along with the questionnaire, course guide, and summary of CA strategies will be sent to each participant via email.

Attached are two documents: 1) the CS1 course guide and 2) a brief description of the CA method. The course guide includes a set of learning outcomes and related instructional strategies that are intended to be used to deliver an 8-week fully online CS1 course. In particular, CA strategies have been incorporated where deemed appropriate to support the course content.

Please review the attached brief description of CA and course guide and then respond to the following Round 1 questionnaire.

Round 1 Questionnaire: CS1 Course Guide Review

Instructions: When evaluating the design of instruction, it is important to consider three criteria: *effectiveness* (i.e., how well do the instructional strategies align with the stated goals and objectives of the course?); *efficiency* (i.e., to what extent is effectiveness accomplished using the least amount of resources possible?); and *appeal* (i.e., how willing are students to become engaged in the learning experience?).

I would like your input on these three criteria as well as the use of CA strategies to support the instruction. After you review the course guide, please complete the following questionnaire. Please be honest and as specific as possible in your comments and recommendations as your input will help me make improvements to the course design.

1. *Effectiveness*: If you were to implement this course as designed, do you think it would be effective in helping students gain the knowledge and skills needed to achieve the stated learning outcomes? Yes/No. If no, please explain.

2. *Efficiency*: If you were to implement this course as designed, do you think you could deliver it in an efficient manner? Yes/No. If no, please explain.

3. *Appeal*: If you were to implement this course as designed, do you think it would be appealing to students? Yes/No. If no, please explain.

4. *Cognitive apprenticeship*: Do you think the cognitive apprenticeship strategies as described within the course guide are appropriate? That is, do the strategies support the type of content that is being taught? Yes/No. If no, please explain.

5. *Additional thoughts*: In addition to your comments above, what specific comments, questions, and/or recommendations do you have for improving the effectiveness, efficiency, and appeal of this course design? What suggestions do you have for improving the clarity of the course guide in general? Please refer to sections and or page numbers where appropriate.

So that the study can be developed in a reasonable amount of time, I would very much appreciate your reply within a week of receiving this note. If you need more time, I would also very much appreciate a brief note from you, so I can adjust the timing for the next round of consultations.

Thanks!

Best regards,

Appendix B

Round 1 -- Computer Science 1 Course Guide with CA Strategies

Course Title: Computer Science 1 (CS1)

Delivery Method: Online

Target Audience: The target audience is the adult, online student, with no previous computer programming experience. The CS1 course is typically the first professional course taken in any of the computer-related programs, such as Computer Science, Information Systems, and Game Design.

Length of Course: The course is designed for an eight-week term, with the last week dedicated exclusively to the final exam. No new material is covered in the eighth week. The course is designed to be challenging and fast-paced, with at least one new topic presented in each of the seven weeks of the course.

Technology/Tools: The course is delivered using several technologies including the following:

1. *Learning Management System (LMS):* eCollege (Pearson eCollege, 2013) is used to house course content such as written lectures, programming problems, quizzes, and the final exam. Online discussions and grading are also contained within the LMS.
2. *Videoconferencing System:* Adobe Connect (Adobe, 2013). Adobe Connect is used for the modeling sessions held every week, where the instructor explains how to develop a program that is conceptually similar to the programming problem that needs to be completed by students that week.
3. *Short Instructional Videos:* Jing (TechSmith, 2013). The instructor has used Jing to prepare a set of about 60 short instructional videos, posted in eCollege, that explain each individual subtopic in detail, including coding samples in Visual Studio.
4. *Short Feedback Videos:* Jing (TechSmith, 2013). The instructor offers individual feedback for the weekly programming problems, where the student's work is briefly analyzed and commented on. The video is accompanied with a rubric-based written feedback as well.

Course Description: CS1 is an introduction to the fundamentals of imperative computer programming using one of the programming languages included within Microsoft Visual Studio 2012. Even though one particular language is used throughout the course (Visual Basic, C++, or C#), the student will be able to apply the skills learned to any programming language. The approach used is hands-on with various computer programming assignments to reinforce the application of fundamental concepts. In order to focus on the foundational topics of computer

programming, object-oriented topics are purposely not covered. This course design is specifically oriented towards the use of Visual Basic as host language.

Required Textbook: Zak, D. (2014). *Programming with Microsoft Visual Basic 2012*, (6th Ed.). Boston, MA: Cengage Learning. ISBN-13: 978-1-285-07792-5, ISBN-10: 1-285-07792-X. This textbook's author has prepared a collection of short instructional videos closely tied to the "lessons" covered in each chapter. The use of this set of instructional videos is included with the purchase of the textbook. This course design is specifically oriented towards one of the versions of CS1 (using Visual Basic as host language.)

Learning Objectives (LOs)

At the end of this course, the learner will:

1. Use Integrated Development Environment (IDE) (i.e., Visual Studio) to:
   1.1. Create, save, compile, and run a program.
   1.2. Design and create a graphical user interface form.
   1.3. Debug a program.
2. Code and test a program that uses variables, arithmetic expressions, and built-in methods.
3. Code and test a program that uses one or more decisions.
4. Code and test a program that requires iteration.
5. Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list.
6. Code and test a program that implements arrays.
7. Code and test a program requiring sequential file input / output.

Course Summary

The week's topics, learning objectives, readings, activities, and assessments in the course can be summarized as follows (points for each assessment indicated, totaling 1000 points):

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points)
Week 1 LOs 1.1 and 1.2 Introduction to Programming	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 2 LOs 2 Data Types, Variables and Expressions	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 3 LOs 3 and 1.3 Decisions and Debugging	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 4 LO 4 Iteration	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 5 LO 5 Modularization	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15)

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points)
		5. Graded learning reflection (10)
Week 6 LO 6 Arrays	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 7 LOs 7 Sequential Files	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (35) 2. Graded discussion topics (20) 3. Quiz (30) 4. Graded exercises (scaffolding) (15) 5. Graded learning reflection (10)
Week 8		1. Final Exam (230)

Instructional Strategies

A. <u>Learning Objective 1.1: Students will use an Integrated Development Environment (IDE) to create, save, compile, and run a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will read Chapter 1 ("An Introduction to Visual Basic") and Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers." Focus is on using the basic functions of the IDE such as create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission.

Students will watch a series of short videos that demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission using Visual Studio.

Generative Strategy/Strategies

Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio.

CA Strategy - Modeling: Using a program named "Disappearing Logo," the instructor facilitates a videoconference session to demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission. The program incorporates a simple graphical interface design with an image and a couple of buttons.

Assessments

Individual Lab Exercise: "Hello World" program. Students will analyze, design, and implement a simple salutation-type program to demonstrate proper use of procedures for creating a basic form, creating and editing the code, compiling and saving a program, testing the finished program, and preparing a program for submission. Students are also provided a step-by-step guide for reference.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the individual programming exercise was approached and what specific coding techniques were used and why.

Quiz: Learners complete a quiz that requires recall of concepts related to the create, save, compile, and run procedure. Examples include (Zak, 2014):

1. Visual Studio is a tool that is generically known as a(n) ____. Answer: IDE, or Integrated Development Environment.
2. What is the name of the IDE window that displays a tree structure of all files and projects in a program? Answer: Solution Explorer window.
3. In Visual Studio, a(n) ____ consists of files and projects. Answer: Solution.
4. Visual Studio uses a pattern to create projects and solutions. This pattern is called a(n) ____. Answer: Template.

Final Exam. Questions that assess the student's ability to create, save, compile, and run a program are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. What is the name of the container in our IDE that contains all files and projects for a complete application? Answer: Solution.
2. True or False: Each object in VB has certain parameters that specify its behavior and appearance. Answer: True.
3. True or False: It is feasible to sort the contents of the Properties window in either alphabetic of numeric order. Answer: False.
4. The window used to create in Visual Studio the graphical user interface for your program is called _____. Answer: Windows Form Designer.

B. <u>Learning Objective 1.2: Students will use an Integrated Development Environment (IDE) to design and create a graphical user interface form and perform basic input and output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers." Focus is on the concepts of creating a form, naming object in a form, adding action buttons, displaying text in a label, and clearing text in an object.

Students will watch a series of short videos that demonstrate these topics.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio to design and create a graphical user interface form and perform basic input and output. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Disappearing Logo," program, the instructor facilitates a videoconference session to demonstrate how to design and create a graphical user interface form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on form design issues, use of various form objects, and event handling.

Assessments

Individual Lab Exercise: "Grading Card" program. Point ranges for each of the letter grades (A, B, C, etc.) are displayed when the user clicks on the appropriate button. "Clear" and "Exit" buttons are also included. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded program problem developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to the design and creation of a graphical user interface form. Examples include (Zak, 2014):

1. The behavior and characteristics of a form object is controlled by its ____. Answer: Properties.
2. To change the title bar of a form, we need to set its _____ property. Answer: Text.
3. What is the assignment operator in VB? Answer: Equal sign (=).
4. Which of the many form objects should we use to display static information to the user? Answer: Label.

Final Exam. Questions that assess the student's ability to create a form, name objects in a form, add action buttons, display text in a label, and clear text in an object are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. The _____ property of the form sets where on the screen it will be displayed when the program is executed. Answer: StartPosition.
2. Which window in the IDE is used to obtain the form objects that we need to place in the form at design time? Answer: Toolbox.
3. The window in the IDE that displays syntax error messages after compilation is called ____. Answer: Error List.
4. What is the correct method to execute so that a VB form is closed normally? Answer: Me.Close().

C. <u>Learning Objective 1.3: Students will use an Integrated Development Environment (IDE) to debug a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Appendix "A" in the textbook ("Finding and Fixing Program Errors"). They will also review the written lecture, "Debugging in Visual Studio." Focus is on the three types of programming errors (syntax, logic, and runtime), how to discover each type of error, and how to use the IDE's debugging tools to discover logic errors.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion are the types of programming errors, how to discover them, and the use of the Visual Studio's debugging tool.

CA Strategy - Modeling: Using as basis the "Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to identify the three types of programming errors and also how to use the IDE's debugging tools to discover logic errors. The "Social Burger" program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

Assessments

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz. Learners completes a quiz that requires them to recall concepts related to the types of programming errors and how to detect and correct them. Examples include (Zak, 2014):

1. When programmers do _____ the errors in a program and its causes are identified, so they can be fixed. Answer: Debugging.
2. What statement will be highlighted in yellow when we are stepping line-by-line in our code within the debugger? Answer: The next statement to be executed.
3. Logic errors are listed in the Error List window, right? Answer: False.
4. What keyboard key is used to Step Into the code when debugging? Answer: F8 key.

Final Exam. Questions that assess the student's ability to identify, detect, and correct the three types of programming errors are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. When using the Step Into tool within the debugger, it is only the variables and properties named in the highlighted line the ones that you can see the values of, right? Answer: False.

2. Which debugging tool is used in order to stop execution of the program and cede control to the Debugger? Answer: The breakpoint tool.
3. Let's suppose that intTotalScore has a value of 200 and that intTests has a value of zero. The statement dblAvg = intTotalScore / intTests will _____ . Answer: Result in a runtime error.
4. Let's suppose that intTotalScore has a value of zero and that intTests has a value of 10. The statement dblAvg = intTotalScore / intTests will _____. Answer: Assign zero to the dblAvg variable.

E. <u>Learning Objective 2: Code and test a program that uses variables, arithmetic expressions, and built-in methods</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 3 ("Using Variables and Constants"). They will also review the written lecture, "Basic Data Types and Variables." Focus is on the concepts of data types, variables, conversion of data types, expressions, and numerical operators.

Students will watch a series of short videos that demonstrate these topics: (1) variables, (2) accepting input, (3) expressions, (4) DateDiff() function, (5) internal program documentation, and (6) MessageBox.Show() method.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on data types, variables, conversion of data types, expressions, and numerical operators. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to use data types, variables, conversion of data types, expressions, and numerical operators. This program requires the acceptance and validation of three grades and then the average is calculated and displayed.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on data types, variables, conversion of data types, expressions, and numerical operators.

Assessments

Individual Lab Exercises: (1) "Dates to Graduation" program. The user supplies name and major plus a graduation date (selected with a DateTimePicker) and the program computes and displays

the days remaining until graduation in a small dialog box in the center of the screen. (2) "Area and Volume Calculator" program: The user supplies the length of a square plot, the radius of a circle, or the radius of a sphere, and the program computes its area or volume, displaying the results. The complete program must be analyzed, designed, and implemented. Students are required to analyze, design, and implement both programs.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to data types, variables, conversion of data types, expressions, and numerical operators. Examples include (Zak, 2014):

1. To change or "cast" the data type of a variable, we can use the _____ class methods in our programs. Answer: Convert.
2. Option Strict On warrants that variables must have a data type declared in the Dim statement. Answer: False.
3. The _____ of the ToString method is used to control how many decimal places we want to display in the resulting string. Answer: formatting string.
4. The _____ constant can be concatenated into a string to force a new line in the text displayed in a control. Answer: ControlChars.NewLine, vbNewLine, vbCR, or vbLF.

Final Exam. Questions that assess the student's ability to declare data types, declare and use variables, convert data types, create expressions, and use numerical operators are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. True or False. The operator used to concatenate two strings is the % operator. Answer: False.
2. True or False. The InputBox() function will always return a text value, even if the user entered nothing when prompted. Answer: False.
3. True or False: Static variables will retain their values even after the procedure where they were declared has terminated. Answer: True.
4. The ____ method is used to convert or "cast" a Date variable into a String. Answer: Date.TryParse.

F. <u>Learning Objective 3: Code and test a program that uses one or more decisions</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapters 4 ("The Selection Structure") and 5 ("More on the Selection Structure"). They will also review the written lecture, "Conditional Statements, Comparison & Logical Operators." Focus is on the concepts of decision statements, Boolean logic, and input validation.

Students will watch a series of short videos that demonstrate these topics: (1) IF statement, (2) Select-Case statement, (3) accepting input, (4) validating string values, (5) validating numeric values, (6) multiple validations, (7) using Radioboxes and Checkboxes, and (8) using Listboxes.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on decision statements, Boolean logic, and input validation. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "The Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to design and write decision statements, use Boolean logic, and validate user's input. This program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on decision statements, Boolean logic, and input validation.

Assessments

Individual Lab Exercise: "Drive-In Income Calculator" program. The program requires input of the type of night (Regular / Car), number of tickets sold, numbers of cars admitted, number of candy sold, and number of popcorn sold, and computes and displays the income generated in a ListBox object. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to decision statements, Boolean logic, and input validation. Examples include (Zak, 2014):

1. To change a string to all-upper characters we can use the _____ method. Answer: ToUpper.
2. True or False: The AndAlso logical operator uses a short-circuit evaluation. Answer: True.
3. True or False: A compound condition using the logical operator OR will be true if either of the individual conditions is true. Answer: True.
4. The < and > operators are called _____ operators. Answer: Logical.

Final Exam. Questions that assess the student's ability to design and write decision statements, use Boolean logic, and validate user's input are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. True or False: It is not possible to specify the icon that the MessageBox.Show() method will display to the user. Answer: False.
2. In a Select-Case statement over variable intCode which of the following Case clauses are valid? Case Is > 7; Case 3, 5; Case 1 To 4; All of them. Answer: All of them.
3. Let's suppose that the Text property of the txtPrice textbox contains the value 75, what value will the Double.TryParse(txtPrice.Text, dblPrice) method return? False; True; 75; 75.0; 75.00; None of them. Answer: True.

G. <u>Learning Objective 4: Code and test a program that requires iteration</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 6 ("The Repetition Structure"). They will also review the written lecture, "Loops." Focus is on the concept of iteration statements.

Students will watch a series of short videos that demonstrate these topics: (1) loop statements, (2) Do-Loop statement, (3) For-Next statement, (4) InputBox() function, (5) using loops in validation – string, and (6) using loops in validation – numeric.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on iteration statements. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to design and use iteration statements. In this

program the user inputs the name, current age, retirement age, current salary, and expected annual raise and the program computes and displays the salary at retirement and the accumulated lifetime earnings.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on iteration statements.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. The user supplies the name and number of seasons played for a hockey player. And then the Goals and Assists for each season played. Input must be done through an InputBox() function. The total Goals, Assists, and Points are then computed and displayed. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to iteration statements. Examples include (Zak, 2014):

1. Is a For-Next statement a pretest or a posttest loop? Answer: Pretest.
2. True or False: A Do-Loop statement can be used to implement a posttest structure. Answer: True.
3. True or False: In flowchart terms, a diamond shape is used to designate a For-Next loop, right? Answer: False.
4. Let's suppose that you create a loop where you ask the user for some numeric input (stored by you in variable intNumericInput after validation) and use that numeric input in the following statement within the loop: intTotal = intTotal + intNumericInput. What type of variable is intTotal? Answer: Accumulator.

Final Exam. Questions that assess the student's ability to design and use iteration statements are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. True or False: We cannot nest a loop within other types of structures, such as If-Then-Else statements. Answer: False.
2. When considering nested loops, the (inner/outer) _____ loop will typically be processed less times than the (inner/outer) _____ loop. Answer: Outer, inner.

3. To immediately stop a Do-Loop we would use the _____ statement. Answer: Exit Do.
4. The _____ listbox property has the position of the item that is currently selected in the listbox. Answer: SelectedIndex.

H. <u>Learning Objective 5: Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 7 ("Sub and Function Procedures"). They will also review the written lecture, "Modularization." Focus is on the concept of modularization.

Students will watch a series of short videos that demonstrate these topics: (1) modularization – invocation, (2) modularization - data transfer, (3) Sub procedures, and (4) Function procedures.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on modularization. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to take a non-modularized program (from a previous week) and modularize it. The Functions implemented are ValidateString(), ValidateInteger, and ValidateDouble(), plus the Sub ComputeResults(), with both ByVal and ByRef parameters.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on modularization.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. Same program developed in a previous week but modularized. Students are required to use at least one Function and one Sub modules, and to pass data in both ByVal and ByRef. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to modularization. Examples include (Zak, 2014):

1. If you are interested in passing only the value of variable to a module, you need to use the ____ keyword when declaring the parameter in the module's declaration.. Answer: ByVal.
2. True or False: A Sub procedure can return a value, but a Function procedure cannot. Answer: False.
3. What statement is used to return a value from a Function procedure? Answer: Return.

Final Exam. Questions that assess the student's ability to modularize a program are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. True or False. The ByVal parameters of a procedure have module scope and are destroyed when the procedure ends. Answer: True.
2. True or False. The following module invocation cannot invoke a ByRef parameter: Call ComputeSalary (1500.25). Answer: True.
3. If you are writing a Function procedure and you want to return the value of variable dblSalary, what statement do you use? Answer: Return dblSalary.

I. <u>Learning Objective 6: Code and test a program that implements arrays</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 9 ("Arrays"). They will also review the written lecture, "Arrays." Focus is on the concept of arrays.

Students will watch a series of short videos that demonstrate these topics: (1) declaring arrays, (2) updating arrays, (3) two-dimensional arrays, (4) parallel arrays, and (5) processing Arrays (computing an average.)

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on arrays. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to declare and use arrays. This program allows the user to supply up to 30 grades in a course's assignments and the program displays all of them in a Listbox and computes the average for the course. Arrays must be used to store the data.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on arrays.

Assessments

Individual Lab Exercise: "Building Rents" program. The user enters the basic data for a building (name and number of floors) and then supplies the rents for all floors. Finally, the user chooses one of the floors and displays the corresponding rent. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to arrays. Examples include (Zak, 2014):

1. What array method would you use to learn the number of elements that the array has? Answer: Length.
2. True or False: If we want to sort an array in reverse order, we would use the Array.Reverse method, since the Array.Sort method will sort the array in the normal, ascending order. Answer: False.
3. True or False: Is the For Each-Next statement capable of processing each element of a one-dimensional array in its entirety? Answer: True.

Final Exam. Questions that assess the student's ability to declare and use arrays are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. In a set of two parallel arrays, how are the values in one array related to the other array's values? Answer: By their index position in the array.
2. The strStates and strCapitals arrays are parallel arrays. If Florida is stored in the second element in the strStates array, where is its capital (Tallahassee) stored? Answer: strCapitals(1).

3. Which of the following statements assigns the string “Florida” to the element located in the third column, fifth row in the two-dimensional strStates array? strStates(3, 5) = "Florida "; strStates(5, 3) = "Florida "; strStates(4, 2) = "Florida "; strStates(2, 4) = "Florida ". Answer: strStates(4, 2) = "Florida ".

J. <u>Learning Objective 7: Code and test a program requiring sequential file input / output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 10 (“Structures and Sequential access Files”). They will also review the written lecture, “Sequential Files.” Focus is on the concept of sequential files.

Students will watch a series of short videos that demonstrate these topics: (1) what is a file, (2) opening and closing files, (3) methods to open a file, (4) reading from a file, (5) writing to a file, (6) CSV files, and (7) file error handling.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on sequential files. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the “Lifetime Earnings” program from a previous week, the instructor facilitates a videoconference session to demonstrate how to add file handling features to it, saving the data as a CSV file for multiple users and then restoring it from the file for display into a second form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are personalized to each student. The instructor posts a series of exercises focused on sequential files.

Assessments

Individual Lab Exercise: “Hockey Player” program. This program is the same developed in previous weeks but expanded to save the data for any number of players. Then a second form is used to display all sets of data thus saved to the file. Students are required to analyze, design, and implement the program.

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that explains how the lab exercise was approached and what specific coding techniques were used and why.

Quiz: Learners completes a quiz that requires them to recall concepts related to sequential files. Examples include (Zak, 2014):

1. Let's suppose that our VB program needs to know if a specific file is present in a hard disk. What method do we use to do it? Answer: Exists.
2. What method can we use to determine if a file has or does not have more data to read from? Answer: Peek().
3. If we want to write to a sequential file, which object do we need to use? Answer: StreamWriter.

Final Exam. Questions that assess the student's ability to use sequential files are included in the final exam given at the end of the semester. Examples include (Zak, 2014):

1. The AppendText method creates a _____ object. Answer: StreamWriter.
2. The Peek method returns _____ when the end of the file is reached. Answer: -1.
3. If the file to be opened exists, the _____ method erases the file's contents. Answer: CreateText.

Appendix C

Round 1 -- Brief Description of Cognitive Apprenticeship

When offered in the face-to-face or blended learning classroom, a CS1 ("Computer Science 1") course taught using cognitive apprenticeship (CA) methods have offered good results, both in performance and retention (Vihavainen, Paksula, Luukkainen, & Kurhila, 2011). CA is derived from the classical style of learning in which many hand crafts are learned: a "master" in the craft teaches the "apprentice" how the craft is developed through modeling, starting from simple items and progressing towards more complex items, while all the time providing support and scaffolding to the apprentice. This scaffolding provided by the master is then gradually removed as the apprentice gains experience.

CA can be implemented in various ways but some of the essential elements are as follows (Collins, Brown, & Newman, 1989):

8. *Modeling*. The instructor offers periodic modeling sessions where the instructor demonstrates the thought processes developed while attempting to solve programming problems. The objective is not to demonstrate features of the computer language used but instead, to show students how the instructor would approach the solution to the problem.
9. *Coaching*. The instructor offer specific guidance to the student while working on exercises so that the correct approach is applied to the solution of the problems, with the objective of correcting performance deviations as soon as possible.
10. *Scaffolding*. Based on gradually more difficult exercises, the instructor leads the student toward ever more complex challenges, until the student reaches the needed learning objectives. As these exercises progress forward, the instructor's support is gradually removed, until at the end of the course practically no support is needed.
11. *Articulation*. Students must explain to the instructor why a specific approach to a problem works, so that the thinking process leading to the solution can be analyzed and solidified in the student's mind.
12. *Reflection*. After each main learning experience, a "reflection" paper is prepared where the students summarizes what they have learned during the previous period and how it can be used in practice.
13. *Exploration*. Students are encouraged to try out new approaches to resolution of the problems, with the intent of developing independent thought.

Given that a significant number of students have difficulties when facing the CS1 course, and given the good results that CA has offered to CS1 courses when taught face-to-face or in a blended classroom, it appears reasonable to determine how well a CA approach will apply to a purely online CS1 course. This topic is interesting not only to the researcher, who has taught this course online for over ten years and continually seeks to improve retention and performance in it, but also for the computer science teaching community at large.

Appendix D

Round 2 -- E-mail to Panel of Experts

Dear Dr. Xxx:

Thank again you for your participation in Round 1 of the study! I have analyzed all the replies from the panel and prepared a set of documents for Round 2, attached. The documents include the following:

1. Document_of_Revisions_Round1ToRound2.docx -- Has a tabular summary of all changes made to the course design as a result of the Round 1 suggestions.

2. Round2-CS1CourseGuideWithCAStrategies-MarkUp.docx -- Is the exact same document from Round 1, but showing the "mark-up" of changes: text deleted (~~sample of text deleted~~) and text added / changed (sample of text added / changed).

3. Round2-CS1CourseGuideWithCAStrategies.docx -- Is the complete, Round 2 document which incorporates all changes suggested.

Following are the instructions for Round 2. Please review the attached documentation and then respond to the following four questions on the Round 2 questionnaire.

Round 2 Questionnaire: Introduction to Computer Programming / CS1 Course Guide Review

Instructions: When evaluating the design of instruction, it is important to consider three criteria: *effectiveness* (i.e., how well do the instructional strategies align with the stated goals and objectives of the course?); *efficiency* (i.e., to what extent is are the goals and objectives of the course accomplished using the least amount of resources possible?); and *appeal* (i.e., how willing are students to become engaged in the learning experience?).

I would like your input on these three criteria as well as the use of CA strategies to support the instruction. After you review the modified course guide, please complete the following questionnaire. Please be honest and as specific as possible in your comments and recommendations as your input will help me make improvements to the course design.

1. *Effectiveness*: The course design is effective (i.e., the learning activities align with the stated goals and objectives of the course):

   1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree.

   Comments/Suggestions for Improvement:

   _____.

2. *Efficiency*: The course design is efficient (i.e., the goals and objectives of the course are achieved using the least amount of resources possible):

   1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree.

   Comments/Suggestions for Improvement:
   _____.

3. *Appeal*: The course design is appealing (i.e., the learning activities are designed to engage students in the learning experience):

   1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree.

   Comments/Suggestions for Improvement:
   _____.

4. *Cognitive apprenticeship*: The cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught):

   1-strongly disagree; 2-disagree; 3-undecided; 4-agree; 5-strongly agree.

   Comments/Suggestions for Improvement:
   _____.

5. *Additional thoughts*: In addition to your comments above, what specific comments, questions, and/or recommendations do you have for improving the effectiveness, efficiency, and appeal of this modified course design? What suggestions do you have for improving the clarity of the course guide in general? Please refer to sections and or page numbers where appropriate.

So that the study can be developed in a reasonable amount of time, I would very much appreciate your reply within two weeks of receiving this note. If you need more time, I would also very much appreciate a brief note from you, so I can adjust the timing for the next round of consultations.

Thanks!

Best regards,

Appendix E

Round 2 -- Document of Revisions

Document of Revisions (DoR)

To: Delphi Study Panel Members
Date: 3/8/2014
From: Reinaldo Fernandez
Re: Round 1 To Round 2, Course Guide changes

Following is a summary of the changes made to the Round 1 Course Guide to incorporate the suggestions made by the panel. The "Number" indicated is included merely as reference, while the "Change Made" details the change made and incorporated in the Round 2 Course Guide, with the column "Rationale" explaining why the change was made.

Number	Change Made	Rationale
1	Course name changed to "Introduction to Computer Programming / CS1".	One panel member believes that some topics (specifically, object-oriented programming) are necessary to be added to the course in order for it to be properly called a "CS1" course (first professional course taken in computer science.) Since discussing if the course is a proper implementation or not of the ACM's official "CS1" course design is not included within the scope of the present study, the name change is offered as an alternative.
2	Course preceded by a so-called "CS0" course.	A preliminary course is convenient as preparation for this course, focusing on algorithms and computer-based problem solving. The course could be described as follows: *This course introduces the basics of programming logic and algorithm design, including use of different data types and variables, expressions, decisions, iterative logic, and modularization. Students learn to design and document programs using tools such as flowcharts and pseudocode.*
3	Required software specified.	The required software should be specified, which is Microsoft Visual Studio 2012 Express Edition (or higher.)
4	Weekly quizzes eliminated.	Several panel members thought that the academic load as originally conceived in the design is too high, particularly for a short-duration course such as this one. Eliminating the weekly quiz --but maintaining the comprehensive Final Exam-- allows retaining all elements of cognitive apprenticeship while reducing some of the academic load.

Number	Change Made	Rationale
5	Reflection essay replaced by Standard Solution Analysis.	Several panel members thought that the Reflection essay, as originally designed, would be somewhat out of place in a computer programming course. As such, one of these panel members offered the alternative idea of doing the reflection (which is required by cognitive apprenticeship) based on the standard solution to the previous week's programming problem, including a comparison of the student's own solution to this standard solution, pointing out and analyzing the differences.
6	Grading rubrics added.	One panel member suggested adding grading rubrics for all programming problems.
7	Minor changes.	Clarification of minor items and correction of obscure passages, as follows: 1. The short instructional videos should be available to students throughout the entire duration of the course. 2. Mention of other computer languages was dropped from the description (only VB is now mentioned.) 3. The "sequential file input / output" designation was replaced by "streamed file input / output" designation, which is more precise. 4. Suggested scheduled due dates added to "Course Summary" section. 5. Written lecture name for learning objectives 1.1 and 1.2 changed to "Problem Solving with Computers / Introduction to Visual Basic / Form Design". 6. The Final Exam includes both multiple-choice questions and "essay"=style questions, where some code needs to be written.

Appendix F

Round 2 -- Computer Science 1 Course Guide with CA Strategies

Course Title: Introduction to Computer Programming / CS1 ("ICP/CS1" for short.)

Delivery Method: Online

Target Audience: The target audience is the adult, online student, with no previous computer programming experience. The ICP/CS1 course is typically the first professional course taken in any of the computer-related programs, such as Computer Science, Information Systems, and Game Design. This course is typically preceded by what is called in the literature a "CS0" course, where basic algorithms and computer-based problem solving is taught.

Length of Course: The course is designed for an eight-week term, with the last week dedicated exclusively to the final exam. No new material is covered in the eighth week. The course is designed to be challenging and fast-paced, with at least one new topic presented in each of the seven weeks of the course.

Technology/Tools: The course is delivered using several technologies including the following:

5. *Learning Management System (LMS):* eCollege (Pearson eCollege, 2013) is used to house course content such as written lectures, programming problems, quizzes, and the final exam. Online discussions and grading are also contained within the LMS.
6. *Videoconferencing System:* Adobe Connect (Adobe, 2013). Adobe Connect is used for the modeling sessions held every week, where the instructor explains how to develop a program that is conceptually similar to the programming problem that needs to be completed by students that week.
7. *Short Instructional Videos:* Jing (TechSmith, 2013). The instructor uses Jing to prepare a set of about 60 short instructional videos, posted in eCollege, that explain each individual subtopic in detail, including coding samples in Visual Studio. These videos are available to students throughout the entire course and can be watched at will when needed.
8. *Short Feedback Videos:* Jing (TechSmith, 2013). The instructor offers individual feedback for the weekly programming problems, where the student's work is briefly analyzed and commented on. The video is accompanied with a rubric-based written feedback as well.

Course Description: ICP/CS1 is an introduction to the fundamentals of imperative computer programming using one of the programming languages included within Microsoft Visual Studio 2012. Even though one particular language is used throughout the course (Visual Basic in this particular case), the student will be able to apply the skills learned to any programming language. The approach used is hands-on with various computer programming assignments to reinforce the

application of fundamental concepts. In order to focus on the foundational topics of computer programming, object-oriented topics are purposely not covered. This course design is specifically oriented towards the use of Visual Basic as host language.

Required Textbook: Zak, D. (2014). *Programming with Microsoft Visual Basic 2012*, (6th Ed.). Boston, MA: Cengage Learning. ISBN-13: 978-1-285-07792-5, ISBN-10: 1-285-07792-X. This textbook's author has prepared a collection of short instructional videos closely tied to the "lessons" covered in each chapter. The use of this set of instructional videos is included with the purchase of the textbook. This course design is specifically oriented towards one of the versions of ICP/CS1 (using Visual Basic as host language.)

Required software: Microsoft Visual Studio 2012 Express Edition. (Any 2012 version will do. The "Express Edition" is a free download from the Microsoft website.)

Learning Objectives (LOs)

At the end of this course, the learner will:

8. Use an Integrated Development Environment (IDE) (i.e., Visual Studio) to:
   8.1. Create, save, compile, and run a program.
   8.2. Design and create a graphical user interface form.
   8.3. Debug a program.
9. Code and test a program that uses variables, arithmetic expressions, and built-in methods.
10. Code and test a program that uses one or more decisions.
11. Code and test a program that requires iteration.
12. Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list.
13. Code and test a program that implements arrays.
14. Code and test a program requiring streamed file input / output.

Course Summary

The week's topics, learning objectives, readings, activities, and assessments in the course can be summarized as follows (points for each assessment indicated, totaling 1000 points) – NOTE: For Round 2 the points were adjusted so as to eliminate the weekly quiz and as such the points were adjusted as follows: Programming problem (50 points each); Graded exercises (20 points each); and Graded learning reflection / solution analysis (20 points each):

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points) – Due date
Week 1 LOs 1.1 and 1.2 Introduction to Programming	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5
Week 2 LOs 2 Data Types, Variables and Expressions	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5 4. Graded learning reflection / solution analysis (20) – Day 7
Week 3 LOs 3 and 1.3 Decisions and Debugging	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5 4. Graded learning reflection / solution analysis (20) – Day 7
Week 4 LO 4 Iteration	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5 4. Graded learning reflection / solution analysis (20) – Day 7
Week 5 LO 5 Modularization	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points) – Due date
		4. Graded learning reflection / solution analysis (20) – Day 7
Week 6 LO 6 Arrays	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5 4. Graded learning reflection / solution analysis (20) – Day 7
Week 7 LOs 7 Sequential Files	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (50) – Day 7 2. Graded discussion topics (20) – Day 3 to Day 7 3. Graded exercises (scaffolding) (20) – Day 5 4. Graded learning reflection / solution analysis (20) – Day 7
Week 8		1. Final Exam (250) – Day 1 to Day 7

Instructional Strategies

A. <u>Learning Objective 1.1: Students will use an Integrated Development Environment (IDE) to create, save, compile, and run a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will read Chapter 1 ("An Introduction to Visual Basic") and Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers / Introduction to Visual Basic / Form Design." Focus is on using the basic functions of the IDE such as create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission.

Students will watch a series of short videos that demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission using Visual Studio.

Generative Strategy/Strategies

Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio.

CA Strategy - Modeling: Using a program named "Disappearing Logo," the instructor facilitates a videoconference session to demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission. The program incorporates a simple graphical interface design with an image and a couple of buttons.

Assessments

Individual Lab Exercise: "Hello World" program. Students will analyze, design, and implement a simple salutation-type program to demonstrate proper use of procedures for creating a basic form, creating and editing the code, compiling and saving a program, testing the finished program, and preparing a program for submission. Students are also provided a step-by-step guide for reference. The grading rubric is as follows (shared with the "Grading Card" program below):

1. Part "A" – Hello World	
Correct building of the program	10%
Correct operation of the program	10%
2. Part "B" – Grading card	
Correct operation	10%
Overall form design	10%
Appropriate labels for all form objects	10%
Correct operation of "Clear" function.	10%
Correct operation of "Exit" function.	10%
Option Strict On set	10%
Internal program documentation	10%
Clean code.	10%
3. Other issues	
N/A	
TOTAL	100%

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Final Exam. Questions that assess the student's ability to create, save, compile, and run a program are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

5. What is the name of the container in our IDE that contains all files and projects for a complete application? Answer: Solution.
6. True or False: Each object in VB has certain parameters that specify its behavior and appearance. Answer: True.
7. True or False: It is feasible to sort the contents of the Properties window in either alphabetic of numeric order. Answer: False.
8. The window used to create in Visual Studio the graphical user interface for your program is called _____. Answer: Windows Form Designer.

B. <u>Learning Objective 1.2: Students will use an Integrated Development Environment (IDE) to design and create a graphical user interface form and perform basic input and output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers / Introduction to Visual Basic / Form Design." Focus is on the concepts of creating a form, naming object in a form, adding action buttons, displaying text in a label, and clearing text in an object.

Students will watch a series of short videos that demonstrate these topics.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio to design and create a graphical user interface form and perform basic input and output. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Disappearing Logo," program, the instructor facilitates a videoconference session to demonstrate how to design and create a graphical user interface form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on form design issues, use of various form objects, and event handling.

Assessments

Individual Lab Exercise: "Grading Card" program. Point ranges for each of the letter grades (A, B, C, etc.) are displayed when the user clicks on the appropriate button. "Clear" and "Exit" buttons are also included. Students are required to analyze, design, and implement the program. The grading rubric is as follows (shared with the "Hello World" program above):

1. Part "A" – Hello World	
Correct building of the program	10%
Correct operation of the program	10%
2. Part "B" – Grading card	
Correct operation	10%
Overall form design	10%
Appropriate labels for all form objects	10%
Correct operation of "Clear" function.	10%
Correct operation of "Exit" function.	10%
Option Strict On set	10%
Internal program documentation	10%
Clean code.	10%
3. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded program problem developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Final Exam. Questions that assess the student's ability to create a form, name objects in a form, add action buttons, display text in a label, and clear text in an object are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

5. The _____ property of the form sets where on the screen it will be displayed when the program is executed. Answer: StartPosition.
6. Which window in the IDE is used to obtain the form objects that we need to place in the form at design time? Answer: Toolbox.
7. The window in the IDE that displays syntax error messages after compilation is called ____. Answer: Error List.
8. What is the correct method to execute so that a VB form is closed normally? Answer: Me.Close().

C. <u>Learning Objective 1.3: Students will use an Integrated Development Environment (IDE) to debug a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Appendix "A" in the textbook ("Finding and Fixing Program Errors"). They will also review the written lecture, "Debugging in Visual Studio." Focus is on the three types of programming errors (syntax, logic, and runtime), how to discover each type of error, and how to use the IDE's debugging tools to discover logic errors.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion are the types of programming errors, how to discover them, and the use of the Visual Studio's debugging tool.

CA Strategy - Modeling: Using as basis the "Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to identify the three types of programming errors and also how to use the IDE's debugging tools to discover logic errors. The "Social Burger" program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

Assessments

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Final Exam. Questions that assess the student's ability to identify, detect, and correct the three types of programming errors are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

5. When using the Step Into tool within the debugger, it is only the variables and properties named in the highlighted line the ones that you can see the values of, right? Answer: False.
6. Which debugging tool is used in order to stop execution of the program and cede control to the Debugger? Answer: The breakpoint tool.
7. Let's suppose that intTotalScore has a value of 200 and that intTests has a value of zero. The statement dblAvg = intTotalScore / intTests will _____ . Answer: Result in a runtime error.
8. Let's suppose that intTotalScore has a value of zero and that intTests has a value of 10. The statement dblAvg = intTotalScore / intTests will _____. Answer: Assign zero to the dblAvg variable.

E. <u>Learning Objective 2: Code and test a program that uses variables, arithmetic expressions, and built-in methods</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 3 ("Using Variables and Constants"). They will also review the written lecture, "Basic Data Types and Variables." Focus is on the concepts of data types, variables, conversion of data types, expressions, and numerical operators.

Students will watch a series of short videos that demonstrate these topics: (1) variables, (2) accepting input, (3) expressions, (4) DateDiff() function, (5) internal program documentation, and (6) MessageBox.Show() method.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on data types, variables, conversion of data types, expressions, and numerical operators. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to use data types, variables, conversion of data types, expressions, and numerical operators. This program requires the acceptance and validation of three grades and then the average is calculated and displayed.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on data types, variables, conversion of data types, expressions, and numerical operators.

Assessments

Individual Lab Exercises: (1) "Dates to Graduation" program. The user supplies name and major plus a graduation date (selected with a DateTimePicker) and the program computes and displays the days remaining until graduation in a small dialog box in the center of the screen. (2) "Area and Volume Calculator" program: The user supplies the length of a square plot, the radius of a circle, or the radius of a sphere, and the program computes its area or volume, displaying the results. The complete program must be analyzed, designed, and implemented. Students are required to analyze, design, and implement both programs. The grading rubric is as follows:

1. Part "A" - Graduation Date	
Correct operation	20%
Handling of data type conversions	10%
"Exit" & "Clear" buttons work OK	3%
Overall form design	3%
Option Strict On set	3%
Internal program documentation	3%
Clean code.	3%
2. Part "B" - Area / Volume Calculator	
Correct operation	20%
Validation of user's input	10%
Handling of data type conversions	10%
"Exit" & "Clear" buttons work OK	3%
Overall form design	3%
Option Strict On set	3%
Internal program documentation	3%
Clean code.	3%
3. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to declare data types, declare and use variables, convert data types, create expressions, and use numerical operators are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

5. True or False. The operator used to concatenate two strings is the % operator. Answer: False.
6. True or False. The InputBox() function will always return a text value, even if the user entered nothing when prompted. Answer: False.
7. True or False: Static variables will retain their values even after the procedure where they were declared has terminated. Answer: True.
8. The ____ method is used to convert or "cast" a Date variable into a String. Answer: Date.TryParse.

F. <u>Learning Objective 3: Code and test a program that uses one or more decisions</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapters 4 ("The Selection Structure") and 5 ("More on the Selection Structure"). They will also review the written lecture, "Conditional Statements, Comparison & Logical Operators." Focus is on the concepts of decision statements, Boolean logic, and input validation.

Students will watch a series of short videos that demonstrate these topics: (1) IF statement, (2) Select-Case statement, (3) accepting input, (4) validating string values, (5) validating numeric values, (6) multiple validations, (7) using Radioboxes and Checkboxes, and (8) using Listboxes.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on decision statements, Boolean logic, and input validation. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "The Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to design and write decision statements, use Boolean logic, and validate user's input. This program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on decision statements, Boolean logic, and input validation.

Assessments

Individual Lab Exercise: "Drive-In Income Calculator" program. The program requires input of the type of night (Regular / Car), number of tickets sold, numbers of cars admitted, number of candy sold, and number of popcorn sold, and computes and displays the income generated in a ListBox object. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

1. Drive-In Revenue	
Correct operation	20%
Validation of user's input	25%
Handling of data type conversions	20%
Overall form design	10%
Option Strict On set	10%
Internal program documentation	10%
Clean code.	5%
2. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to design and write decision statements, use Boolean logic, and validate user's input are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

4. True or False: It is not possible to specify the icon that the MessageBox.Show() method will display to the user. Answer: False.
5. In a Select-Case statement over variable intCode which of the following Case clauses are valid? Case Is > 7; Case 3, 5; Case 1 To 4; All of them. Answer: All of them.
6. Let's suppose that the Text property of the txtPrice textbox contains the value 75, what value will the Double.TryParse(txtPrice.Text, dblPrice) method return? False; True; 75; 75.0; 75.00; None of them. Answer: True.

G. <u>Learning Objective 4: Code and test a program that requires iteration</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 6 ("The Repetition Structure"). They will also review the written lecture, "Loops." Focus is on the concept of iteration statements.

Students will watch a series of short videos that demonstrate these topics: (1) loop statements, (2) Do-Loop statement, (3) For-Next statement, (4) InputBox() function, (5) using loops in validation – string, and (6) using loops in validation – numeric.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on iteration statements. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to design and use iteration statements. In this program the user inputs the name, current age, retirement age, current salary, and expected annual raise and the program computes and displays the salary at retirement and the accumulated lifetime earnings.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on iteration statements.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. The user supplies the name and number of seasons played for a hockey player. And then the Goals and Assists for each season played. Input must be done through an InputBox() function. The total Goals, Assists, and Points are then computed and displayed. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

1. Hockey Statistics	
Correct operation	20%
Validation of user's input (w/loops)	30%
Handling of data type conversions	20%
Overall form design	5%
Option Strict On set	10%
Internal program documentation	10%
Clean code.	5%
2. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to design and use iteration statements are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

5. True or False: We cannot nest a loop within other types of structures, such as If-Then-Else statements. Answer: False.
6. When considering nested loops, the (inner/outer) _____ loop will typically be processed less times than the (inner/outer) _____ loop. Answer: Outer, inner.
7. To immediately stop a Do-Loop we would use the _____ statement. Answer: Exit Do.
8. The _____ listbox property has the position of the item that is currently selected in the listbox. Answer: SelectedIndex.

H. Learning Objective 5: Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 7 ("Sub and Function Procedures"). They will also review the written lecture, "Modularization." Focus is on the concept of modularization.

Students will watch a series of short videos that demonstrate these topics: (1) modularization – invocation, (2) modularization - data transfer, (3) Sub procedures, and (4) Function procedures.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on modularization. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to take a non-modularized program (from a previous week) and modularize it. The Functions implemented are ValidateString(), ValidateInteger, and ValidateDouble(), plus the Sub ComputeResults(), with both ByVal and ByRef parameters.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on modularization.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. Same program developed in a previous week but modularized. Students are required to use at least one Function and one Sub modules, and to pass data in both ByVal and ByRef. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

1. Hockey Player Statistics	
Correct operation	10%
Validation of user's input	10%
Handling of data type conversions	10%
Correct modularization	50%
Overall form design	5%
Option Strict On set	5%
Internal program documentation	5%
Clean code.	5%
2. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to modularize a program are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

4. True or False. The ByVal parameters of a procedure have module scope and are destroyed when the procedure ends. Answer: True.
5. True or False. The following module invocation cannot invoke a ByRef parameter: Call ComputeSalary (1500.25). Answer: True.
6. If you are writing a Function procedure and you want to return the value of variable dblSalary, what statement do you use? Answer: Return dblSalary.

I. <u>Learning Objective 6: Code and test a program that implements arrays</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 9 ("Arrays"). They will also review the written lecture, "Arrays." Focus is on the concept of arrays.

Students will watch a series of short videos that demonstrate these topics: (1) declaring arrays, (2) updating arrays, (3) two-dimensional arrays, (4) parallel arrays, and (5) processing Arrays (computing an average.)

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on arrays. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to declare and use arrays. This program allows the user to supply up to 30 grades in a course's assignments and the program displays all of them in a Listbox and computes the average for the course. Arrays must be used to store the data.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on arrays.

Assessments

Individual Lab Exercise: "Building Rents" program. The user enters the basic data for a building (name and number of floors) and then supplies the rents for all floors. Finally, the user chooses one of the floors and displays the corresponding rent. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

1. Building Rents	
Correct operation	15%
Validation of user's input	15%
Handling of data type conversions	10%
Correct use of arrays	35%
Overall form design	5%
Option Strict On set	5%
Internal program documentation	10%
Clean code.	5%
2. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to declare and use arrays are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

4. In a set of two parallel arrays, how are the values in one array related to the other array's values? Answer: By their index position in the array.
5. The strStates and strCapitals arrays are parallel arrays. If Florida is stored in the second element in the strStates array, where is its capital (Tallahassee) stored? Answer: strCapitals(1).
6. Which of the following statements assigns the string "Florida" to the element located in the third column, fifth row in the two-dimensional strStates array? strStates(3, 5) = "Florida "; strStates(5, 3) = "Florida "; strStates(4, 2) = "Florida "; strStates(2, 4) = "Florida ". Answer: strStates(4, 2) = "Florida ".

J. <u>Learning Objective 7: Code and test a program requiring streamed file input / output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 10 ("Structures and Sequential access Files"). They will also review the written lecture, "Sequential Files." Focus is on the concept of sequential files.

Students will watch a series of short videos that demonstrate these topics: (1) what is a file, (2) opening and closing files, (3) methods to open a file, (4) reading from a file, (5) writing to a file, (6) CSV files, and (7) file error handling.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on sequential files. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Lifetime Earnings" program from a previous week, the instructor facilitates a videoconference session to demonstrate how to add file handling features to it, saving the data as a CSV file for multiple users and then restoring it from the file for display into a second form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on sequential files.

Assessments

Individual Lab Exercise: "Hockey Player" program. This program is the same developed in previous weeks but expanded to save the data for any number of players. Then a second form is used to display all sets of data thus saved to the file. Students are required to analyze, design, and implement the program. Given that this program requires use of all topics covered in the course, including streamed input/output, this program can be considered a "Final Project". The grading rubric is as follows:

1. Hockey Player Statistics	
Correct operation	5%
Validation of user's input	5%
Handling of data type conversions	5%
Data correctly added to file	20%
"Summary" form / menu well designed	20%
"Summary" data correctly displayed	25%
"Clear File" operation correctly done	15%
Option Strict On set / Clean code / Internal program documentation	5%
2. Other issues	
N/A	
TOTAL	100%

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week.

Learning Reflection / Solution Analysis Essay (CA Strategy – Articulation). Students are required to submit a short essay at the end of the week that analyzes the standard solution to the previous week's programming problem (posted by the instructor together with the previous week's feedback) explaining how the lab exercise was approached and what specific coding techniques were used and why. The student also compares their own coding techniques with that standard solution and explains the differences.

Final Exam. Questions that assess the student's ability to use sequential files are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

4. The AppendText method creates a _____ object. Answer: StreamWriter.
5. The Peek method returns _____ when the end of the file is reached. Answer: -1.
6. If the file to be opened exists, the _____ method erases the file's contents. Answer: CreateText.

Appendix G

Round 3 -- E-mail to Panel of Experts

Dear Dr. Xxx:

Thank you for your participation in Round 2 of the study! I have analyzed all the replies from the panel and prepared a set of documents for Round 3, attached. The documents include the following:

1. Document_of_Revisions_Round2ToRound3.docx -- Has a tabular summary of all changes made to the course design as a result of the Round 2 suggestions.

2. Round3-CS1CourseGuideWithCAStrategies-MarkUp.docx -- Is the exact same document from Round 2, but showing the "mark-up" of changes: text deleted (~~sample of text deleted~~) and text added / changed (sample of text added / changed).

3. Round3-CS1CourseGuideWithCAStrategies.docx -- Is the complete, Round 3 document which incorporates all changes suggested.

Following are the instructions for Round 3. Please review the attached documentation and then respond to the following two questions on the Round 3 questionnaire.

Round 3 Questionnaire: Introduction to Computer Programming / CS1 Course Guide Review

Instructions: The goal of the study was to construct and validate an Introduction to Computer Programming / CS1 course that incorporates CA strategies and that is designed for online delivery. The purpose of the first two rounds was to get your input on how the course needed to be revised to meet this goal, and as such in this third and final round I would like to obtain agreement from you and all panel members on the following questions:

To the best of your knowledge:

1. Do you agree that the cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught)?

2. Do you agree that the proposed guide is sufficient for the delivery of the course in an online environment?

3. If you disagree with 1 and/or 2, please explain what additional revisions to the course guide are necessary in order to meet these two objectives.

4. Please provide any additional comments or questions you might have.

So that the study can be developed in a reasonable amount of time, I would very much appreciate your reply within one week of receiving this note. If you need more time, I would also very much appreciate a brief note from you, so I can adjust the timing for the next round of consultations.

Thanks!

Best regards,

Appendix H

Round 3 -- Document of Revisions

Document of Revisions (DoR)

To: Delphi Study Panel Members
Date: 4/14/2014
From: Reinaldo Fernandez
Re: Round 2 To Round 3, Course Guide changes

Following is a summary of the changes made to the Round 2 Course Guide to incorporate the suggestions made by the panel. The "Number" indicated is included merely as reference, while the "Change Made" details the change made and incorporated in the Round 3 Course Guide, with the column "Rationale" explaining why the change was made.

Number	Change Made	Rationale
1	"Suggested Class Policies" section added to the introductory pages.	One panel member opined that --due to the intense pace of the course-- special care must be dedicated to "at-risk" students (those who are not doing well in the course.) Another panel member opined that --also due to the intense pace of the course-- class discussion should be de-emphasized, so that more time can be devoted to the programming problems and the individual forum discussions with the instructor.
2	"Suggested Class Procedures" section added to the introductory pages.	Following the advice of a panel member, it should be clarified that the rubrics included in the Course Design document should be posted in the classroom. In addition, it should also be clarified that the test question samples included in the Course Design document should not be included in the classroom.
3	"Specific instances of Cognitive Apprenticeship implementation" section added to the introductory pages.	Clarifies how each of the six elements of which Cognitive Apprenticeship is composed is mapped to course design features.
4	Graded learning reflection / solution analysis essay eliminated.	Several panel members thought that the academic load as originally conceived in the design is too high, particularly for a short-duration course such as this one. Eliminating the graded learning reflection essay --but emphasizing in its place the individual discussions between student and instructor-- allows retaining all elements of cognitive apprenticeship while reducing some of the academic load.
5	Individual Forum Programming Problem discussion Individual Forum Programming Problem discussion expanded.	The scope and description of this forum was expanded to cover the reflection note required at the end of the week.

Number	Change Made	Rationale
6	Points in the Course Summary were adjusted.	The points for each assessment were modified so as to comply with the material eliminated.
7	Grading rubrics expanded.	The grading rubrics were expanded so they could be more useful to students.
8	Minor changes.	Clarification of minor items and correction of obscure passages, as follows: 7. The description of the individual exercises was clarified. 8. Clarification that the important topic of debugging (LO 1.3) is covered.

Appendix I

Round 3 -- Computer Science 1 Course Guide with CA Strategies

Course Title: Introduction to Computer Programming / CS1 ("ICP/CS1" for short.)

Delivery Method: Online

Target Audience: The target audience is the adult, online student, with no previous computer programming experience. The ICP/CS1 course is typically the first professional course taken in any of the computer-related programs, such as Computer Science, Information Systems, and Game Design. This course is typically preceded by what is called in the literature a "CS0" course, where basic algorithms and computer-based problem solving is taught.

Length of Course: The course is designed for an eight-week term, with the last week dedicated exclusively to the final exam. No new material is covered in the eighth week. The course is designed to be challenging and fast-paced, with at least one new topic presented in each of the seven weeks of the course.

Technology/Tools: The course is delivered using several technologies including the following:

9. *Learning Management System (LMS):* eCollege (Pearson eCollege, 2013) is used to house course content such as written lectures, programming problems, quizzes, and the final exam. Online discussions and grading are also contained within the LMS.
10. *Videoconferencing System:* Adobe Connect (Adobe, 2013). Adobe Connect is used for the modeling sessions held every week, where the instructor explains how to develop a program that is conceptually similar to the programming problem that needs to be completed by students that week.
11. *Short Instructional Videos:* Jing (TechSmith, 2013). The instructor uses Jing to prepare a set of about 60 short instructional videos, posted in eCollege, that explain each individual subtopic in detail, including coding samples in Visual Studio. These videos are available to students throughout the entire course and can be watched at will when needed.
12. *Short Feedback Videos:* Jing (TechSmith, 2013). The instructor offers individual feedback for the weekly programming problems, where the student's work is briefly analyzed and commented on. The video is accompanied with a rubric-based written feedback as well. Before the end of the following week, this feedback should be discussed with the student privately in the student's Individual Forum (as noted in sections "*CA Exploration - Individual Forum Programming Problem discussion*" below.)

Course Description: ICP/CS1 is an introduction to the fundamentals of imperative computer programming using one of the programming languages included within Microsoft Visual Studio 2012. Even though one particular language is used throughout the course (Visual Basic in this particular case), the student will be able to apply the skills learned to any programming language.

The approach used is hands-on with various computer programming assignments to reinforce the application of fundamental concepts. In order to focus on the foundational topics of computer programming, object-oriented topics are purposely not covered. This course design is specifically oriented towards the use of Visual Basic as host language.

Required Textbook: Zak, D. (2014). *Programming with Microsoft Visual Basic 2012*, (6th Ed.). Boston, MA: Cengage Learning. ISBN-13: 978-1-285-07792-5, ISBN-10: 1-285-07792-X. This textbook's author has prepared a collection of short instructional videos closely tied to the "lessons" covered in each chapter. The use of this set of instructional videos is included with the purchase of the textbook. This course design is specifically oriented towards one of the versions of ICP/CS1 (using Visual Basic as host language.)

Required Software: Microsoft Visual Studio 2012 Express Edition. (Any 2012 version will do. The "Express Edition" is a free download from the Microsoft website.)

Suggested Class Policies: Although these ultimately depend on the institution where the course will be implemented, there are some policies that are suggested for best results with this course design:

1. "At-risk" students should be identified early in the class and offered help. The nature of the topics in this course, which build upon one another, imply that failure in one topic will cause problems for the student in the next topic, so early detection and help are recommended. The instructor should keep close tabs on how well students are doing in the course, particularly with the programming problems, and take action as soon as poor performance is detected.
2. Class discussions should be de-emphasized, though not eliminated, so that more time can be devoted to the scaffolding exercises and also to the individual programming problem discussions with the instructor, both developed within the Individual Forum.

Suggested Class Procedures: The following procedures are suggested for better development of this course:

1. The rubrics indicated in the sections below should be posted in the classroom, so that students know with more precision what is expected of them and how the programming problems will be graded.
2. The sample test questions indicated in the sections below should not be posted in the classroom, so as to avoid student confusion.

Specific instances of Cognitive Apprenticeship implementation. As described in an attached document, Cognitive apprenticeship (CA) consists of the following elements (immediately following each of the elements, it is specified how that particular element is implemented in the course design, in dark red typography):

14. *Modeling*. The instructor offers periodic modeling sessions where the instructor demonstrates the thought processes developed while attempting to solve programming problems. The objective is not to demonstrate features of the computer language used but instead, to show students how the instructor would approach the solution to the problem. *Implementation*. Instructor-led videoconference session.

15. *Coaching*. The instructor offer specific guidance to the student while working on exercises so that the correct approach is applied to the solution of the problems, with the objective of correcting performance deviations as soon as possible.
    Implementation. Individual asynchronous exercise sessions.
16. *Scaffolding*. Based on gradually more difficult exercises, the instructor leads the student toward ever more complex challenges, until the student reaches the needed learning objectives. As these exercises progress forward, the instructor's support is gradually removed, until at the end of the course practically no support is needed.
    Implementation. Individual asynchronous exercise sessions.
17. *Articulation*. Students must explain to the instructor why a specific approach to a problem works, so that the thinking process leading to the solution can be analyzed and solidified in the student's mind.
    Implementation. Individual Forum Programming Problem discussion.
18. *Reflection*. After each main learning experience, a "reflection" exercise is developed where the student summarizes what they have learned during the previous period and how it can be used in practice.
    Implementation. Individual Forum Programming Problem discussion.
19. *Exploration*. Students are encouraged to try out new approaches to resolution of the problems, with the intent of developing independent thought.
    Implementation. Discussion forum participation; and also Individual Forum Programming Problem discussion.

Learning Objectives (LOs)

At the end of this course, the learner will:

15. Use an Integrated Development Environment (IDE) (i.e., Visual Studio) to:
    15.1. Create, save, compile, and run a program.
    15.2. Design and create a graphical user interface form.
    15.3. Debug a program.
16. Code and test a program that uses variables, arithmetic expressions, and built-in methods.
17. Code and test a program that uses one or more decisions.
18. Code and test a program that requires iteration.
19. Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list.
20. Code and test a program that implements arrays.
21. Code and test a program requiring streamed file input / output.

Course Summary

The week's topics, learning objectives, readings, activities, and assessments in the course can be summarized as follows (points for each assessment indicated, totaling 1000 points) – NOTE: For Round 3 the points were adjusted so as to reduce the points for the Graded discussion topics and to reflect the elimination of the Graded learning reflection, so the points were adjusted as follows: Programming problem (60 points each); Graded discussion topics (10); Graded exercises (40 points each); and Final Exam (230 points.)

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points) – Due date
Week 1 LOs 1.1 and 1.2 Introduction to Programming	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 2 LOs 2 Data Types, Variables and Expressions	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 3 LOs 3 and 1.3 Decisions and Debugging	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 4 LO 4 Iteration	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 5 LO 5 Modularization	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 6 LO 6 Arrays	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7

Week, LOs, and Topics	Readings/Class Preparation	Activities/Assignments (points) – Due date
		3. Graded exercises (scaffolding) (40) – Day 5
Week 7 LOs 7 Sequential Files	1. Textbook chapter(s) 2. Written lecture (eCollege) 3. Short videos (Jing) 4. Videoconference (Adobe Connect)	1. Programming problem (60) – Day 7 2. Graded discussion topics (10) – Day 3 to Day 7 3. Graded exercises (scaffolding) (40) – Day 5
Week 8		1. Final Exam (230) – Day 1 to Day 7

Instructional Strategies

A. <u>Learning Objective 1.1: Students will use an Integrated Development Environment (IDE) to create, save, compile, and run a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will read Chapter 1 ("An Introduction to Visual Basic") and Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers / Introduction to Visual Basic / Form Design." Focus is on using the basic functions of the IDE such as create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission.

Students will watch a series of short videos that demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission using Visual Studio.

Generative Strategy/Strategies

Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio.

CA Strategy - Modeling: Using a program named "Disappearing Logo," the instructor facilitates a videoconference session to demonstrate how to create a basic form, create and edit the code, compile and save a program, test the finished program, and prepare a program for submission. The program incorporates a simple graphical interface design with an image and a couple of buttons.

Assessments

Individual Lab Exercise: "Hello World" program. Students will analyze, design, and implement a simple salutation-type program to demonstrate proper use of procedures for creating a basic form, creating and editing the code, compiling and saving a program, testing the finished program, and preparing a program for submission. Students are also provided a step-by-step guide for reference. The grading rubric is as follows (shared with the "Grading Card" program below):

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Part "A" – Hello World					
Correct building of the program	10%	Program correctly constructed.	Acceptable construction but with some issues.	Unacceptable, very limited construction.	Not built.
Correct operation of the program	10%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
2. Part "B" – Grading card					
Correct operation	10%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Overall form design	10%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Appropriate labels for all form objects	10%	Good labels used.	Some objects left unlabeled or with improper labeling.	Most objects left unlabeled or with improper labeling.	No labeling done.
Correct operation of "Clear" function	10%	"Clear" correctly implemented.	Only a partial "Clear" done.	Practically no objects cleared.	No objects cleared.
Correct operation of "Exit" function	10%	"Exit" correctly implemented.	Done but with a method different from Close().	-	Not included.
Option Strict On set	10%	Set.	-	-	Not set.
Internal program documentation	10%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	10%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to create, save, compile, and run a program are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

9. What is the name of the container in our IDE that contains all files and projects for a complete application? Answer: Solution.
10. True or False: Each object in VB has certain parameters that specify its behavior and appearance. Answer: True.
11. True or False: It is feasible to sort the contents of the Properties window in either alphabetic of numeric order. Answer: False.
12. The window used to create in Visual Studio the graphical user interface for your program is called _____. Answer: Windows Form Designer.

B. <u>Learning Objective 1.2: Students will use an Integrated Development Environment (IDE) to design and create a graphical user interface form and perform basic input and output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 2 ("Designing Applications"). They will also review the written lecture, "Problem Solving with Computers / Introduction to Visual Basic / Form Design." Focus is on the concepts of creating a form, naming object in a form, adding action buttons, displaying text in a label, and clearing text in an object.

Students will watch a series of short videos that demonstrate these topics.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on the use of Visual Studio to design and create a graphical user interface form and perform basic input and output. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Strategy - Modeling: Using the "Disappearing Logo," program, the instructor facilitates a videoconference session to demonstrate how to design and create a graphical user interface form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on form design issues, use of various form objects, and event handling.

Assessments

Individual Lab Exercise: "Grading Card" program. Point ranges for each of the letter grades (A, B, C, etc.) are displayed when the user clicks on the appropriate button. "Clear" and "Exit" buttons are also included. Students are required to analyze, design, and implement the program. The grading rubric is as follows (shared with the "Hello World" program above):

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Part "A" – Hello World					
Correct building of the program	10%	Program correctly constructed.	Acceptable construction but with some issues.	Unacceptable, very limited construction.	Not built.
Correct operation of the program	10%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
2. Part "B" – Grading card					
Correct operation	10%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Overall form design	10%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Appropriate labels for all form objects	10%	Good labels used.	Some objects left unlabeled or with improper labeling.	Most objects left unlabeled or with improper labeling.	No labeling done.
Correct operation of "Clear" function	10%	"Clear" correctly implemented.	Only a partial "Clear" done.	Practically no objects cleared.	No objects cleared.
Correct operation of "Exit" function	10%	"Exit" correctly implemented.	Done but with a method different from Close().	-	Not included.
Option Strict On set	10%	Set.	-	-	Not set.
Internal program documentation	10%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	10%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded program problem developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to create a form, name objects in a form, add action buttons, display text in a label, and clear text in an object are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

9. The _____ property of the form sets where on the screen it will be displayed when the program is executed. Answer: StartPosition.

10. Which window in the IDE is used to obtain the form objects that we need to place in the form at design time? Answer: Toolbox.
11. The window in the IDE that displays syntax error messages after compilation is called ____. Answer: Error List.
12. What is the correct method to execute so that a VB form is closed normally? Answer: Me.Close().

C. <u>Learning Objective 1.3: Students will use an Integrated Development Environment (IDE) to debug a program</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Appendix "A" in the textbook ("Finding and Fixing Program Errors"). They will also review the written lecture, "Debugging in Visual Studio." Focus is on the three types of programming errors (syntax, logic, and runtime), how to discover each type of error, and how to use the IDE's debugging tools to discover logic errors. This topic of debugging is also covered in other weeks, as the need arises during the modeling sessions or class discussions.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion are the types of programming errors, how to discover them, and the use of the Visual Studio's debugging tool.

CA Strategy - Modeling: Using as basis the "Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to identify the three types of programming errors and also how to use the IDE's debugging tools to discover logic errors. The "Social Burger" program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

Assessments

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to identify, detect, and correct the three types of programming errors are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

9. When using the Step Into tool within the debugger, it is only the variables and properties named in the highlighted line the ones that you can see the values of, right? Answer: False.
10. Which debugging tool is used in order to stop execution of the program and cede control to the Debugger? Answer: The breakpoint tool.
11. Let's suppose that intTotalScore has a value of 200 and that intTests has a value of zero. The statement dblAvg = intTotalScore / intTests will _____ . Answer: Result in a runtime error.
12. Let's suppose that intTotalScore has a value of zero and that intTests has a value of 10. The statement dblAvg = intTotalScore / intTests will _____. Answer: Assign zero to the dblAvg variable.

E. <u>Learning Objective 2: Code and test a program that uses variables, arithmetic expressions, and built-in methods</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 3 ("Using Variables and Constants"). They will also review the written lecture, "Basic Data Types and Variables." Focus is on the concepts of data types, variables, conversion of data types, expressions, and numerical operators.

Students will watch a series of short videos that demonstrate these topics: (1) variables, (2) accepting input, (3) expressions, (4) DateDiff() function, (5) internal program documentation, and (6) MessageBox.Show() method.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on data types, variables, conversion of data types, expressions, and numerical operators. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to use data types, variables, conversion of data types, expressions, and numerical operators. This program requires the acceptance and validation of three grades and then the average is calculated and displayed.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on data types, variables, conversion of data types, expressions, and numerical operators.

Assessments

Individual Lab Exercises: (1) "Dates to Graduation" program. The user supplies name and major plus a graduation date (selected with a DateTimePicker) and the program computes and displays the days remaining until graduation in a small dialog box in the center of the screen. (2) "Area

and Volume Calculator" program: The user supplies the length of a square plot, the radius of a circle, or the radius of a sphere, and the program computes its area or volume, displaying the results. The complete program must be analyzed, designed, and implemented. Students are required to analyze, design, and implement both programs. The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Part "A" - Graduation Date					
Correct operation	20%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Handling of data type conversions	10%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Overall form design	3%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	3%	Set.	-	-	Not set.
Internal program documentation	3%	Good program ID header, module headers, in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	3%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
2. Part "B" - Area / Volume Calculator					
Correct operation	20%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input	10%	Good validation used, such as with TryParse() method.	Acceptable validation but not too complete.	Poor validation that causes runtime or other errors if the user enters invalid data.	No validation at all.
Handling of data type conversions	10%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Overall form design	3%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	3%	Set.	-	-	Not set.
Internal program documentation	3%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	3%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to declare data types, declare and use variables, convert data types, create expressions, and use numerical operators are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

9. True or False. The operator used to concatenate two strings is the % operator. Answer: False.
10. True or False. The InputBox() function will always return a text value, even if the user entered nothing when prompted. Answer: False.
11. True or False: Static variables will retain their values even after the procedure where they were declared has terminated. Answer: True.
12. The ____ method is used to convert or "cast" a Date variable into a String. Answer: Date.TryParse.

F. <u>Learning Objective 3: Code and test a program that uses one or more decisions</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapters 4 ("The Selection Structure") and 5 ("More on the Selection Structure"). They will also review the written lecture, "Conditional Statements, Comparison & Logical Operators." Focus is on the concepts of decision statements, Boolean logic, and input validation.

Students will watch a series of short videos that demonstrate these topics: (1) IF statement, (2) Select-Case statement, (3) accepting input, (4) validating string values, (5) validating numeric values, (6) multiple validations, (7) using Radioboxes and Checkboxes, and (8) using Listboxes.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on decision statements, Boolean logic, and input validation. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "The Social Burger," program, the instructor facilitates a videoconference session to demonstrate how to design and write decision statements, use Boolean logic, and validate user's input. This program is a menu-taking program for a small restaurant, with multiple decisions necessary to implement the needed logic.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on decision statements, Boolean logic, and input validation.

Assessments

Individual Lab Exercise: "Drive-In Income Calculator" program. The program requires input of the type of night (Regular / Car), number of tickets sold, numbers of cars admitted, number of candy sold, and number of popcorn sold, and computes and displays the income generated in a ListBox object. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Drive-In Income Calculator					
Correct operation	20%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input	25%	Good validation used, checking (1) Entry; (2) Data type: and (3) Range.	Acceptable validation but not too complete.	Poor validation that causes runtime or other errors if the user enters invalid data.	No validation of any kind made.
Handling of data type conversions	20%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Overall form design	10%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	10%	Set.	-	-	Not set.
Internal program documentation	10%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	5%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to design and write decision statements, use Boolean logic, and validate user's input are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

7. True or False: It is not possible to specify the icon that the MessageBox.Show() method will display to the user. Answer: False.
8. In a Select-Case statement over variable intCode which of the following Case clauses are valid? Case Is > 7; Case 3, 5; Case 1 To 4; All of them. Answer: All of them.
9. Let's suppose that the Text property of the txtPrice textbox contains the value 75, what value will the Double.TryParse(txtPrice.Text, dblPrice) method return? False; True; 75; 75.0; 75.00; None of them. Answer: True.

G. <u>Learning Objective 4: Code and test a program that requires iteration</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 6 ("The Repetition Structure"). They will also review the written lecture, "Loops." Focus is on the concept of iteration statements.

Students will watch a series of short videos that demonstrate these topics: (1) loop statements, (2) Do-Loop statement, (3) For-Next statement, (4) InputBox() function, (5) using loops in validation – string, and (6) using loops in validation – numeric.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on iteration statements. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to design and use iteration statements. In this program the user inputs the name, current age, retirement age, current salary, and expected annual raise and the program computes and displays the salary at retirement and the accumulated lifetime earnings.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on iteration statements.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. The user supplies the name and number of seasons played for a hockey player. And then the Goals and Assists for each season played. Input must be done through an InputBox() function. The total Goals, Assists, and Points are then computed and displayed. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Hockey Statistics					
Correct operation	20%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input (with loops.)	30%	Good validation used, checking (1) Entry; (2) Data type: and (3) Range. Correct use of loops.	Acceptable validation but not too complete or incorrect use of loops.	Acceptable validation nut without using loops at all.	No validation of any kind made.
Handling of data type conversions	20%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Overall form design	5%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	10%	Set.	-	-	Not set.
Internal program documentation	10%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	5%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to design and use iteration statements are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

9. True or False: We cannot nest a loop within other types of structures, such as If-Then-Else statements. Answer: False.
10. When considering nested loops, the (inner/outer) _____ loop will typically be processed less times than the (inner/outer) _____ loop. Answer: Outer, inner.
11. To immediately stop a Do-Loop we would use the _____ statement. Answer: Exit Do.
12. The _____ listbox property has the position of the item that is currently selected in the listbox. Answer: SelectedIndex.

H. <u>Learning Objective 5: Code and test a program that uses modular design and implements one or more modules that obtains data passed to it through its parameter list</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 7 ("Sub and Function Procedures"). They will also review the written lecture, "Modularization." Focus is on the concept of modularization.

Students will watch a series of short videos that demonstrate these topics: (1) modularization – invocation, (2) modularization - data transfer, (3) Sub procedures, and (4) Function procedures.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on modularization. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "Lifetime Earnings," program, the instructor facilitates a videoconference session to demonstrate how to take a non-modularized program (from a previous week) and modularize it. The Functions implemented are ValidateString(), ValidateInteger, and ValidateDouble(), plus the Sub ComputeResults(), with both ByVal and ByRef parameters.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on modularization.

Assessments

Individual Lab Exercise: "Hockey Statistics" program. Same program developed in a previous week but modularized. Students are required to use at least one Function and one Sub modules, and to pass data in both ByVal and ByRef. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Hockey Statistics					
Correct operation	10%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input	10%	Good validation used, checking (1) Entry; (2) Data type: and (3) Range.	Acceptable validation but not too complete.	Poor validation that causes runtime or other errors if the user enters invalid data.	No validation of any kind made.
Handling of data type conversions	10%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Correct modularization	50%	Correct use of both Functions and Sub procedures. Correct passing of both ByVal and ByRef parameters.	Correct use of both Functions and Sub procedures but no ByRef parameters used.	Limited use of Functions and / or Sub procedures. Limited or non-existent passing of parameters.	No modularization done.
Overall form design	5%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	5%	Set.	-	-	Not set.
Internal program documentation	5%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	5%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to modularize a program are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

7. True or False. The ByVal parameters of a procedure have module scope and are destroyed when the procedure ends. Answer: True.
8. True or False. The following module invocation cannot invoke a ByRef parameter: Call ComputeSalary (1500.25). Answer: True.
9. If you are writing a Function procedure and you want to return the value of variable dblSalary, what statement do you use? Answer: Return dblSalary.

I. <u>Learning Objective 6: Code and test a program that implements arrays</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 9 ("Arrays"). They will also review the written lecture, "Arrays." Focus is on the concept of arrays.

Students will watch a series of short videos that demonstrate these topics: (1) declaring arrays, (2) updating arrays, (3) two-dimensional arrays, (4) parallel arrays, and (5) processing Arrays (computing an average.)

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on arrays. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "Grade Average Calculator," program, the instructor facilitates a videoconference session to demonstrate how to declare and use arrays. This program allows the user to supply up to 30 grades in a course's assignments and the program displays all of them in a Listbox and computes the average for the course. Arrays must be used to store the data.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on arrays.

Assessments

Individual Lab Exercise: "Building Rents" program. The user enters the basic data for a building (name and number of floors) and then supplies the rents for all floors. Finally, the user chooses one of the floors and displays the corresponding rent. Students are required to analyze, design, and implement the program. The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Building Rents					
Correct operation	15%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input	15%	Good validation used, checking (1) Entry; (2) Data type: and (3) Range.	Acceptable validation but not too complete.	Poor validation that causes runtime or other errors if the user enters invalid data.	No validation of any kind made.
Handling of data type conversions	10%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Correct use of arrays	35%	Arrays correctly declared and used.	Acceptable declaration and use of arrays but with issues.	Arrays correctly declared but not used.	No arrays used.
Overall form design	5%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	5%	Set.	-	-	Not set.
Internal program documentation	10%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	5%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to declare and use arrays are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

7. In a set of two parallel arrays, how are the values in one array related to the other array's values? Answer: By their index position in the array.
8. The strStates and strCapitals arrays are parallel arrays. If Florida is stored in the second element in the strStates array, where is its capital (Tallahassee) stored? Answer: strCapitals(1).
9. Which of the following statements assigns the string "Florida" to the element located in the third column, fifth row in the two-dimensional strStates array? strStates(3, 5) =

"Florida "; strStates(5, 3) = "Florida "; strStates(4, 2) = "Florida "; strStates(2, 4) = "Florida ". Answer: strStates(4, 2) = "Florida ".

J. <u>Learning Objective 7: Code and test a program requiring streamed file input / output</u>.

Content Type/Performance Level: Concepts/Recall and also Cognitive Procedure/Application

Initial Presentation: Students will review Chapter 10 ("Structures and Sequential access Files"). They will also review the written lecture, "Sequential Files." Focus is on the concept of sequential files.

Students will watch a series of short videos that demonstrate these topics: (1) what is a file, (2) opening and closing files, (3) methods to open a file, (4) reading from a file, (5) writing to a file, (6) CSV files, and (7) file error handling.

Generative Strategy/Strategies

CA Exploration - Discussion Forum Participation: Students will participate in asynchronous discussion forums that are guided by the instructor. The topic of discussion reinforces the information presented in the readings, written lecture, and videos. Emphasis will be placed on sequential files. The instructor challenges the student in open class discussion through alternative or novel coding techniques.

CA Exploration and Reflection- Individual Forum Programming Problem discussion: Each student will participate in asynchronous discussion in private with the instructor, discussing the previous week's programming problem and the instructor's grading of it. The student's coding and design choices are analyzed and compared with the instructor's own suggested solution. The objective is to perform a critical evaluation of any issue that may have been incurred and explore alternative options. The student is tasked with closing the week's discussions with a note where the student summarizes what they have learned during the previous period and how it can be used in practice.

CA Strategy - Modeling: Using the "Lifetime Earnings" program from a previous week, the instructor facilitates a videoconference session to demonstrate how to add file handling features to it, saving the data as a CSV file for multiple users and then restoring it from the file for display into a second form.

CA Strategy - Coaching and Scaffolding: The instructor creates an exercise session using the asynchronous forum. These sessions are individualized for each student. The instructor posts a series of exercises focused on sequential files.

Assessments

Individual Lab Exercise: "Hockey Player" program. This program is the same developed in previous weeks but expanded to save the data for any number of players. Then a second form is used to display all sets of data thus saved to the file. Students are required to analyze, design, and implement the program. Given that this program requires use of all topics covered in the course, including streamed input/output, this program can be considered a "Final Project". The grading rubric is as follows:

		Valuation			
Item	**%**	**100%**	**70%**	**30%**	**0%**
1. Xxx					
Correct operation	5%	Operates as specified.	Operates as specified except for some aspects.	Has serious operational issues.	Program doesn't compile.
Validation of user's input	5%	Good validation used, checking (1) Entry; (2) Data type: and (3) Range.	Acceptable validation but not too complete.	Poor validation that causes runtime or other errors if the user enters invalid data.	No validation of any kind made.
Handling of data type conversions	5%	All data types are converted when needed.	Only some data types are converted but not all.	Nearly no data type conversions.	No data type conversions at all.
Data correctly added to file	20%	Data added in an appropriate format.	Data added but with an inefficient or inappropriate format.	Only pre-formatted lines of data added.	No data added to the file.
"Summary" form / menu well designed	20%	Form and menu correctly designed.	Acceptable but not as required design of form and/or menu.	Poor form and menu design.	"Summary" form not included.
"Summary" data correctly displayed	25%	Data correctly displayed.	Some data displayed but not in a good format. Or incomplete data shown.	Only pre-formatted lines of data displayed.	"Summary" data not displayed.
"Clear File" operation correctly done	15%	Operation correctly implemented.	Form cleared but not the file.	-	"Clear File" operation not done.
Overall form design	1.25%	Good design.	Too much wasted space / No form title.	Poor aesthetics.	No form included.
Option Strict On set	1.25%	Set.	-	-	Not set.
Internal program documentation	1.25%	Good program ID header, module headers, and in-text comments.	One of the three elements missing.	Two of the three elements missing.	No documentation included.
Clean code.	1.25%	Good, easy to read code.	Acceptable but not great readability.	Poor readability.	No code included.
TOTAL	100%				

CA Strategy – Reflection - Feedback on Graded Assignment: A short five-minute video is prepared with feedback from the instructor for each graded individual lab exercise developed. Frank but constructive feedback is provided and the student is encouraged to continue dialogue with the instructor via the individual discussion forum.

General Discussion Forums. A minimum number of substantive posts over different days are required each week, according to the each particular institution's class participation policy.

Final Exam. Questions that assess the student's ability to use sequential files are included in the final exam given at the end of the semester. The exam includes multiple-choice questions and "essay"-type questions where some code has to be written. Examples include (Zak, 2014):

7. The AppendText method creates a _____ object. Answer: StreamWriter.
8. The Peek method returns _____ when the end of the file is reached. Answer: -1.
9. If the file to be opened exists, the _____ method erases the file's contents. Answer: CreateText.

Appendix J

Round 3 -- Final E-mail to Panel of Experts

Dear Dr. Xxx:

Thank you for your participation in the third and final round of the study! I have analyzed all the replies from the panel and I can report that there was consensus that the course design as proposed in the final round does comply with the objectives of the internal validation, that is:

1. The cognitive apprenticeship strategies as described within the course guide are appropriate (i.e., the strategies support the type of content that is being taught.)

2. The proposed guide is sufficient for the delivery of the course in an online environment.

Needless to say, I am immensely grateful for your participation in the study and I very much appreciate your detailed analysis and suggestions, which have made the course design significantly better, both as an implementation of cognitive apprenticeship in the online environment and also with respect to the important parameters of effectiveness, efficiency, and appeal.

If interested in the actual dissertation report that I will write based on this research, please send me a note and I will E-mail it to you once it is approved, later this year.

Thanks!

Best regards,

Appendix K

Nova Southeastern University Institutional Review Board Approval

NOVA SOUTHEASTERN UNIVERSITY
Office of Grants and Contracts
Institutional Review Board

MEMORANDUM

To: Reinaldo Fernandez

From: Ling Wang, Ph.D.
Institutional Review Board

Date: Jan. 13, 2014

Re: *A Cognitive Apprenticeship Approach for Teaching Abstract and Complex Skills in an Online Learning Environment*

IRB Approval Number: wang12151303

I have reviewed the above-referenced research protocol at the center level. Based on the information provided, I have determined that this study is exempt from further IRB review. You may proceed with your study as described to the IRB. As principal investigator, you must adhere to the following requirements:

1) CONSENT: If recruitment procedures include consent forms these must be obtained in such a manner that they are clearly understood by the subjects and the process affords subjects the opportunity to ask questions, obtain detailed answers from those directly involved in the research, and have sufficient time to consider their participation after they have been provided this information. The subjects must be given a copy of the signed consent document, and a copy must be placed in a secure file separate from de-identified participant information. Record of informed consent must be retained for a minimum of three years from the conclusion of the study.

2) ADVERSE REACTIONS: The principal investigator is required to notify the IRB chair and me (954-262-5369 and 954-262-2020 respectively) of any adverse reactions or unanticipated events that may develop as a result of this study. Reactions or events may include, but are not limited to, injury, depression as a result of participation in the study, life-threatening situation, death, or loss of confidentiality/anonymity of subject. Approval may be withdrawn if the problem is serious.

3) AMENDMENTS: Any changes in the study (e.g., procedures, number or types of subjects, consent forms, investigators, etc.) must be approved by the IRB prior to implementation. Please be advised that changes in a study may require further review depending on the nature of the change. Please contact me with any questions regarding amendments or changes to your study.

The NSU IRB is in compliance with the requirements for the protection of human subjects prescribed in Part 46 of Title 45 of the Code of Federal Regulations (45 CFR 46) revised June 18, 1991.

Cc: Protocol File

3301 College Avenue • Fort Lauderdale, FL 33314-7796 • (954) 262-5369
Fax: (954) 262-3977 • Email: *inga@nsu.nova.edu* • Web site: www.nova.edu/cwis/ogc

References

AAUP. (2006). 1940 Statement of Principles on Academic Freedom and Tenure of the American Association of University Professors, Tenth Ed. [PDF document]. Retrieved from the AAUP website: http://www.aaup.org/report/1940-statement-principles-academic-freedom-and-tenure.

ACM. (2001). Computing curricula 2001. *Journal on Educational Resources in Computing, 1*(3).

Adobe. (2013). Adobe Connect [computer software]. San Jose, CA.

Allan, V. & Kolesar, M. (1997). Teaching computer science: a problem solving approach that works. *ACM SIGCUE Outlook, 25*(1-2), 2-10.

Allen, I. & Seaman, J. (2014). *Grade Change: Tracking Online Education in the United States, 2013*. Retrieved from http://sloanconsortium.org/publications/survey/grade-change-2013.

Ambrosio, A., Moreira, F., Almeida, L., Franco, A., & Macedo, J. (2011). Identifying cognitive abilities to improve CS1 outcome. *Proceedings of the 41st ASEE/IEEE Frontiers in Education Conference*. Rapid City, SD, F3G1-F3G7.

Austing, R., Barnes, B., Bonnette, D., Engel, G., & Stokes, G. (1979). Curriculum '78: recommendations for the undergraduate program in computer science -- a report of the ACM curriculum committee on computer science. *Communications of the ACM, 22*(3).

Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., & Kingston, J. (2000). Problem-Based learning for foundation computer science courses. *Computer Science Education, 10*(2), 109-128.

Barker, L.J., McDowell, C., and Kalahar, K. (2009). Exploring Factors that Influence Computer Science Introductory Course Students to Persist in the Major. *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education.* Chattanooga, Tennessee, March 2009.

Barnes, T., Richter, H., Powell, E., Chaffin, A., & Godwin, A. (2007). Game2Learn: building CS1 learning games for retention. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. Dundee, Scotland*, 121-125.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Beier, M., Miller, L., & Wang, S. (2012). Science games and the development of scientific possible selves. *Cultural Studies of Science Education, 7*(4), 963-978.

Bennedsen, J. & Caspersen, M. (2005). Revealing the programming process. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 186-190.

Bennedsen, J. & Caspersen, M. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin, 39*(2), 32-36.

Bowles, K. (1978). A CS1 course based on stand-alone microcomputers. *Proceedings of the 1978 SIGCSE Technical Symposium on Computer Science Education*, 125-127.

Bransford, J. D., & Stein, B. S. (1993). *The IDEAL problem solver: A guide for improving thinking, learning, and creativity (2nd ed.)* New York: W. H. Freeman and Company.

Brown, J., Collins, A. and Duguid, P. (1989) Situated cognition and the culture of learning, *Educational Researcher, 18*(1), 32–42.

Camtasia. (2013). Camtasia Studio [Computer software]. Okemos, MI: TechSmith, Corp. Retrieved March 16, 2013, from http://www.techsmith.com/.

Carlisle, M., Wilson, T., Humphries, J., & Hadfield, S. (2004). Raptor: Introducing programming to non-majors with flowcharts. *Journal of Computing Sciences in Colleges, 19*(4), 52-60.

Collins, A., Brown, J., & Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In *Knowing, Learning and Instruction: Essays in honor of Robert Glaser*. Hillside.

Collins, A., J. Brown, & A. Holum. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 15* (3): 6-41.

Cooper, S., Dann, W., Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Proceedings of the fifth annual CCSC northeastern conference on the Journal of Computing in Small Colleges (CCSC '00)*, 107-116.

Costelloe, E., Sherry, E., & Magee, P. (2009) Experiences Gained Using a Set of SCORM Compliant Reusable Learning Objects for Teaching Programming. *International Journal on E-Learning, 8*(2), 175-191.

Dalkey, N. & Helmer, O. (1963). An experimental application of the Delphi method to the use of experts. *Management Science, 9*(3), 458-467.

Dasarathy, B., Sullivan, K., Schmidt, D. C., Fisher, D. H., & Porter, A. (2014). The past, present, and future of MOOCs and their relevance to software engineering. *Proceedings of the 36th International Conference on Software Engineering*, 212-224.

Demetriadis, S., Papadopoulos, P., Stamelos, I., & Fischer, F. (2008). The effect of scaffolding students' context-generating cognitive activity in technology-enhanced case-based learning. *Computers & Education, 51*(2), 939-954.

Dierbach, C., Taylor, B., Zhou, H., & Zimand, I. (2005). Experiences with a CS0 course targeted for CS1 success. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 317-320.

Doering, A. & Veletsianos, G. (2008). What lies beyond effectiveness and efficiency? Adventure learning design. *Internet and Higher Education, 11*(3-4), 137-144.

Fjuk, A., Berge, O., Bennedsen, J., & Caspersen, M. E. (2004). Learning object-orientation through ICT-mediated apprenticeship. *Proceedings of the Fourth IEEE International Conference on Advanced Learning Technologies (ICALT'04)*, 380-384.

Foster, A. (2008). Games and motivation to learn science: Personal identity, applicability, relevance, and meaningfulness. *Journal of Interactive Learning Research, 19*, 597–614.

Gannon, R., Ley, K., Crawford, C., & Warner, A. (2009). Motivators and inhibitors for university faculty in distance and e-learning. *British Journal of Educational Technology, 40*(1), 149–163.

Garcia, H., Sánchez, E., & Acuña, S. R. (2013). Support for self-regulation in learning complex topics from multimedia explanations: Do learners need extensive or minimal support? *Instructional Science, 41*(3), 539-553.

Gardner, H. (1983). *Frames of mind: The theory of multiple intelligences*. New York: Basic Books.

Garlick, R. & Cankaya, E. (2010). Using Alice in CS1 – A quantitative experiment. *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE'10)*, 165-168.

Ge, X., & Land, S. M. (2004). A conceptual framework for scaffolding ill-structured problem-solving processes using question prompts and peer interactions. *Educational Technology Research and Development*, *52*(2), 5-22.

Gonzalez, G. (2006). A systematic approach to active and cooperative learning in CS1 and its effects on CS2. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 133-137.

Greening, T. (1998). Scaffolding for success in problem-based learning. *Medical Education Online, 3*. Retrieved from http://med-ed-online.net/index.php/meo/issue/view/375

Bouta, H. & Paraskeva, F. (2013). The cognitive apprenticeship theory for the teaching of mathematics in an online 3D virtual environment. *International Journal of Mathematical Education in Science and Technology, 44*(2), 159-178.

Hassinen, M. and Mayra, H. (2006). Learning programming by programming: A case study. *Proceedings of the 6th Baltic Sea conference on computing education research*, 117–119.

Herrmann, N., Popyack, J., Char, B., & Zoski, P. (2004). Assessment of a course redesign: introductory computer programming using online modules. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 66-70.

Hughes, J. and Ramanee-Peiris, D. (2006). ASSISTing CS1 students to learn: learning approaches and object-oriented programming. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 275-279.

Keijonen, H., Kurhila, J., & Vihavainen, A. (2013). Carry-on effect in extreme apprenticeship. *Proceedings of the IEEE Frontiers in Education Conference, 2013*, 1150 - 1155.

Kirschner, P. & Sweller, J. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75-86.

Knobelsdorf, M., Kreitz, C., & Bohne, S. (2014). Teaching theoretical computer science using a cognitive apprenticeship approach. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 67-72.

Kumar, D. (2013). The changing, not evolving pedagogy of CS1. *ACM Inroads, 4*(3), 36-37.

Kuo, F., Hwang, G., Chen, S., & Chen, S. (2012). A cognitive apprenticeship approach to facilitating web-based collaborative problem solving. *Educational Technology & Society, 15*(4), 319-331.

Linstone, H. A., & Turoff, M. (Eds.). (1975). *The Delphi method*. Reading, MA: Addison-Wesley.

Mayer, R., Mautone, P., & Prothero, W. (2002). Pictorial aids for learning by doing in a multimedia geology simulation game. *Journal of Educational Psychology, 94*(1), 171-185.

Mitroff, I. & Turoff, M. (1975). Philosophical and methodological foundations of Delphi. In *The Delphi method* (Chapter 11.B). Reading, MA: Addison-Wesley.

Morey, A. (2004). Globalization and the emergence of for-profit higher education. *Higher Education, 48*(1), 131-150.

Moritz, S., Wei, F., Parvez, S., & Blank, G. (2005). From objects-first to design-first with multimedia and intelligent tutoring. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 99-103.

Morrison, G.R., Ross, S.M., Kalman, H.K. & Kemp, J.E. (2011). *Designing Effective Instruction,* (6th Ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Mow, I., Wing, K., & Yates, G. (2006). The impact of CABLE on teaching computer programming. *Learning by Effective Utilization of Technologies: Facilitating Intercultural Understanding*, Mizoguchi, R. et al. (Eds.) IOS Press. 55-62.

Neville, A. (2008) Problem-based learning and medical education forty years on: A review of its effects on knowledge and clinical performance. *Medical Principles and Practice, 18*(1), 1-9.

Neuman, D. (1995). High school students' use of databases: results of a national Delphi study. *Journal of the American Society for Information Science, 46*(4), 284-298.

Pea, R. (2004). The social and technological dimensions of scaffolding and related theoretical concepts for learning, education, and human activity. *The Journal of the Learning Sciences, 13*(3), 423–451.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *Working group reports on ITiCSE on Innovation and technology in computer science education (ITiCSE-WGR '07)*, Janet Carter and June Amillo (Eds.). ACM, New York, NY, USA, 204-223.

Pearson. (2014). Pearson eCollege [computer software]. Centennial, CO.

Price, K. (2013). *Using visual technologies in the introductory programming courses for Computer Science majors*. Unpublished doctoral dissertation, Nova Southeastern University, Fort Lauderdale, FL.

Phillips, D. C. and Soltis, J. F. (2004). *Perspectives on Learning* (4th Ed.). New York: Teachers College Press, 41-52.

Ramdass, D. (2012). The role of cognitive apprenticeship in learning science in a virtual world. *Cultural Studies of Science Education, 7*(4), 985-992.

Ramo, J. & Vikberg, T. (2014). Extreme apprenticeship – Engaging undergraduate students on a mathematics course. *Proceedings of the 2014 Frontiers in Mathematics and Science Education Research Conference*, 26-33.

Reid, P. (2014). Categories for barriers to adoption of instructional technologies. *Education and Information Technologies, 19*(2), 383-407.

Reigeluth, C. M., & Frick, T. W. (1999). Formative Research: A Methodology for Creating and Improving Design Theories. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models* (633-651). Mahwah, NJ: Lawrence Erlbaum Assoc.

Restrepo, J. & Trefftz, H. (2005). Telepresence support for synchronous distance education. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 63-67.

Rich, L., Perry, H., & Guzdial, M. (2004). A CS1 course designed to address interests of women, *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 190-194.

Richey, R.C. (2005). Validating instructional design and development models. In J.M. Spector, C. Ohrazda, A. Van Schaack, and D.A. Wiley (Eds.), *Innovations in Instructional Technology* (171-185). Mahwah, NJ: Lawrence Erlbaum Assoc.

Richey, R.C. & Klein, J.D. (2007). *Design and Development Research*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.

Robins, A. (2010). Learning edge momentum: A new account of outcomes in CS1. *Computer Science Education, 20*(1), 37-71.

Rosenberg, M. (2003). Redefining e-learning. *Performance Improvement, 42*(3), 38-41.

Rountree, N., Rountree, J., Robins, A., & Hannah, R. (2004). Interacting factors that predict success and failure in a CS1 course. *Proceedings of the Annual Joint Conference Integrating Technology into Computer Science Education*, 101-104.

Soh, L. (2006). Incorporating an intelligent tutoring system into CS1. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 486-490.

Stevenson, D. and Wagner, P. (2006). Developing real-world programming assignments for CS1. *SIGCSE Bulletin – ACM, 38*(3): 158-162.

Stone, J. & Clark, T. (2011). The impact of problem-oriented animated learning modules in a CS1-style course. *Proceedings of the 42nd SIGSCE Technical Symposium on Computer Science Education*, 51-56.

TechSmith. (2013). Jing [computer software]. Okemos, MI.

Thrun, S. (October, 2012). *Democratizing higher education*. Unpublished paper presented at The 18th Annual Sloan Consortium International Conference on Online Learning, October 10-12, Lake Buena Vista, FL.

Tierney, W. (2011). Too big to fail: The role of for-profit colleges and universities in American higher education. *The Magazine of Higher Learning, 43*(6), 27-32.

Tillmann, N., de Halleux, J., Xie, T., Gulwani, S., & Bishop, J. (2013). Teaching and learning programming and software engineering via interactive gaming. *Proceedings of the 35th International Conference on Software Engineering Education*, 1117-1126.

Toff, R. (1955). Are we overprotecting our children in school, too? *The Educational Forum, 19*(2), 165-167.

Tracey, M. (2007). Design and development research: a model validation case. *Educational Technology Research & Development, 57*(4), 553-571

Tracey, M. & Richey, R. (2007). ID model construction and validation: A multiple intelligences case. *Educational Technology Research & Development, 55*(4), 369-390.

Tsang, J., Naughton, P., Leong, S., Hill, A., Kelly, C., & Leahy, A. (2008). Virtual reality simulation in endovascular surgical training. *The Royal Colleges of Surgeons of Edinburgh and Ireland, 6*, 214–220.

Turner, E., Albert, E., Turner, R., & Latour, L. (2007). Retaining majors through the introductory sequence. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, 24-28.

Valentine, D. W. (2004). CS educational research: a meta-analysis of SIGCSE technical symposium proceedings. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 255-259.

Vincenti, G., Braman, J., & Hilberg, J. (2012). Teaching introductory programming through reusable learning objects: A pilot study. *Journal of Computing Sciences in Colleges, 28*(3), 38-45.

Vilner, T., Zue, E., & Sagi, R. (2012). Integrating Video Components in CS1. *Proceedings of the 43rd ACM SIGCSE Technical Symposium on Computer Science Education.* Raleigh, North Carolina, 123-128.

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011) Extreme apprenticeship method in teaching programming for beginners. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 93-98

Vihavainen, A., Paksula, M., Luukkainen, M. & Kurhila, J. (2011). Extreme Apprenticeship method: Key practices and upward scalability. *Proceedings of the 16th ACM Annual Conference on Innovation and Technology in Computer Science Education*. Darmstadt, Germany, 273-277.

Vihavainen, A., Luukkainen, M. & Kurhila, J. (2012). Multi-faceted Support for MOOC in Programming. *Proceedings of the 13th ACM 13th Annual Conference on Information Technology Education*, 171-176.

Vihavainen, A., Vikberg, T., Luukkainen, M., & Partel, M. (2013) Scaffolding students' learning using Test My Code. *Proceedings of the 18th Annual Conference on Innovation and Technology in Computer Science Education*, 117-122

Von Merrienboer, J. & Sweller, J. (2005). Cognitive load theory and complex learning: Recent development and future directions. *Educational Psychology Review, 17*(2), 147-177.

Wang, C., Dong, L., Li, C., Zhang, W., & He, J. (2012). The reform of programming teaching based on constructivism. *Advances in Electric and Electronics*. W. Hu (Ed.) Springer-Verlag, Berlin-Heidelberg, 425-431.

Wass, R. Harland, T., and Mercer, A. (2011). Scaffolding critical thinking in the zone of proximal development. *Higher Education Research & Development, 30*(3), 317–328.

Williams, L., Wiebe, E., Yang. K., Ferzli, M. & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197-212.

Wortman, D. & Rheingans, P. (2007). Visualizing trends in student performance across computer science courses. *Proceedings of the 38th SIGCSE Technical Symposium on Computer science Education*, 430-434.

York, C. & Ertmer, P. (2011). Towards an understanding of instructional design heuristics: an exploratory Delphi study. *Educational Technology Research and Development, 59*(6), 841-863.

Yoo, J., Yoo, S., Seo, S., Dong, Z., & Pettey, C. (2012). Can we teach algorithm development skills? *Proceedings of the 50th Southeast Conference of the ACM*, Tuscaloosa, AL, 101-105.

Yuliya, C., Zingaro, D., & Petersen, A. (2014). Identifying challenging CS1 concepts in a large problem dataset. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 695-700.

Zak, D. (2014). *Programming with Microsoft Visual Basic 2012, (6th Ed.)* Boston, MA: Cengage Learning.