Cognitively Flexible Hypertext in an Object-Oriented Programming Course: Effects of Case-Based Instructional Support on Student Learning

by

Cecil P. Schmidt

B.S., Kansas State University, 1984

M.S., The Wichita State University, 1993

AN ABSTRACT OF A DISSERTATION

Submitted in partial fulfillment of the

Requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Secondary Education

College of Education

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2005

Abstract

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases would improve a student's understanding of the more difficult concepts in an object-oriented programming course. Also investigated were the relationships between the dependent variable student performance with independent variables of motivation, background knowledge, and student attitudes towards the semi-automated reasoning tool. Subjects for the study were randomly assigned from two sections of an introductory object-oriented programming course at an NCAA, Division II university in the Midwest region of the United States. Posttests were used to measure the effects of the semi-automated reasoning tool on learner competency. Background knowledge was collected through student transcripts. Motivation and student attitudes data were collected from surveys. All data were collected during the Spring 2005 semester. Data were analyzed at the $p < .05$ level of significance using a Wilcoxon Signed Ranks test, mixed-design ANOVA, Mann-Whitney U test, Spearman rho correlations, and thematic analysis as well as other statistical techniques.

Results of the study indicated a significant difference between the group who used the semi-automated reasoning tool on complex questions and the group who did not. No significant difference was indicated between the groups on simple questions. A strong positive correlation was indicated between background knowledge and the total test scores for both content areas tested. Results of the correlational analysis between motivation and learner competencies indicated that the type of motivation, be it intrinsic or extrinsic, plays a minimal role in how students performed in this online course.

Finally, students overwhelmingly felt that the semi-automated reasoning tool was an effective instructional-support tool.

Results of this study suggested recommendations for practice as well as for further research. Recommendations for practice include the need for effective use of course management systems, supporting complex content through examples, using performance on background coursework when considering an online course covering complex topics, providing a case-based instructional aid for complex topics, and minimizing the economic costs in using a case-based instructional aid. Recommendations for future research include more research on relationships between background coursework and online courses, effects of a case-based instructional aid on face-to-face courses, development of overarching examples containing content from multiple computer science courses, improvements to the CBJava framework, and extending the framework to other disciplines.

Cognitively Flexible Hypertext in an Object-Oriented Programming Course: Effects of Case-Based Instructional Support on Student Learning

by

Cecil P. Schmidt

B.S., Kansas State University, 1984

M.S., The Wichita State University, 1993

A DISSERTATION

Submitted in partial fulfillment of the

Requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Secondary Education

College of Education

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2005

Approved by:

Major Professor

Dr. Diane McGrath

COGNITIVELY FLEXIBLE HYPERTEXT IN AN OBJECT-ORIENTED PROGRAMMING COURSE: EFFECTS OF CASE-BASED INSTRUCTIONAL SUPPORT ON STUDENT LEARNING

CECIL P. SCHMIDT

2005

Abstract

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases would improve a student's understanding of the more difficult concepts in an object-oriented programming course. Also investigated were the relationships between the dependent variable student performance with independent variables of motivation, background knowledge, and student attitudes towards the semi-automated reasoning tool. Subjects for the study were randomly assigned from two sections of an introductory object-oriented programming course at an NCAA, Division II university in the Midwest region of the United States. Posttests were used to measure the effects of the semi-automated reasoning tool on learner competency. Background knowledge was collected through student transcripts. Motivation and student attitudes data were collected from surveys. All data were collected during the Spring 2005 semester. Data were analyzed at the $p < .05$ level of significance using a Wilcoxon Signed Ranks test, mixed-design ANOVA, Mann-Whitney U test, Spearman rho correlations, and thematic analysis as well as other statistical techniques.

Results of the study indicated a significant difference between the group who used the semi-automated reasoning tool on complex questions and the group who did not. No significant difference was indicated between the groups on simple questions. A strong positive correlation was indicated between background knowledge and the total test scores for both content areas tested. Results of the correlational analysis between motivation and learner competencies indicated that the type of motivation, be it intrinsic or extrinsic, plays a minimal role in how students performed in this online course.

Finally, students overwhelmingly felt that the semi-automated reasoning tool was an effective instructional-support tool.

Results of this study suggested recommendations for practice as well as for further research. Recommendations for practice include the need for effective use of course management systems, supporting complex content through examples, using performance on background coursework when considering an online course covering complex topics, providing a case-based instructional aid for complex topics, and minimizing the economic costs in using a case-based instructional aid. Recommendations for future research include more research on relationships between background coursework and online courses, effects of a case-based instructional aid on face-to-face courses, development of overarching examples containing content from multiple computer science courses, improvements to the CBJava framework, and extending the framework to other disciplines.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincerest appreciation to the people who have offered their assistance and support throughout my program of study. Words cannot express the gratitude for the professionalism, guidance, and support that my major professor, Dr. Diane McGrath, gave me throughout my program. She is truly an educator. Additionally I would like to thank the members of my committee, Dr. Chandima Cumaranatunge, Dr. Tweed Ross, Dr. Donna Lalonde, and my final exam chairperson Dr. Briana Goff, for their time and effort in reading and critiquing the manuscript. In particular I would like to especially thank Dr. Lalonde for her efforts and support. I am in debt to all.

Additionally, I would like to thank members of the Computer Information Sciences department at Washburn University. Especially to Dr. Nancy Tate who supported my entry into the graduate program at KSU and to Dr. Robert Boncella; I guess it is my turn to “pay it forward”.

There will always be a special place in my heart for the KDD group and Dr. William Hsu of the Computer Sciences department at KSU. The two years that I spent working with him and his research group aided me tremendously in focusing on the line of research that I used in this dissertation. It was the study of machine learning which turned my interest into human learning. With Dr. Diane McGrath’s help, I was able to bridge the two together.

Finally, I would like to thank my wife Eileen and my two children Miranda and Jacob. It is difficult to put into words how deeply I appreciate their support, encouragement, and love that they have given me through my studies. I am forever in their debt.

Chapter I: Introduction

Purpose

How might a college level online course in object-oriented programming be developed so that it will be at least as effective as a face-to-face course? Can we provide a learner-centered environment that scaffolds a student's understanding by providing examples that are indexed and linked through hypertext in order to assist in his/her learning? Are there differences in the level of understanding between a student who has taken the online object-oriented programming course and a student who has taken the face-to-face object-oriented programming course? What role does student motivation and background knowledge play in his/her performance and understanding in the online course versus the face-to-face course? This study and the supporting course and instructional aid development address those questions.

The primary purpose of this study was to determine if a student's understanding of the more difficult concepts in an object-oriented programming course would improve by using a semi-automated reasoning tool that provides a set of searchable cases. Specifically, this study tested the learning effects of a semi-automated reasoning tool as an instructional aid in an online object-oriented programming course. The lecture content included digital videos of the instructor's lecture integrated with PowerPoint slides and covered the identical lecture material as in a face-to-face introductory object-oriented programming course. The course content included lecture slides, comments, and code examples that would typically arise in face-to-face teacher-learner interaction. The semi-automated reasoning tool provided integrated instructional content through the use of hypertext including exploratory capabilities with a database of examples. Additionally the semi-automated reasoning tool allowed both the teacher and the learner to add, update, and delete examples in the database. A secondary purpose of the study was to examine the relationships between the dependent variable, student performance, with the independent variables of motivation, background knowledge, and student attitudes towards the case-based hypertext learning tool. Both quantitative and qualitative data were collected and analyzed.

Background

Course Content and Interaction

The cost of developing an online course in terms of a teacher's time and technical knowledge can be quite significant (Smulders, 2004). Therefore it is important to look for methods, tools, and techniques that can minimize this cost. From a larger perspective we acknowledge that developing the content is but one component of a successful distance education course. Moore (1989) suggests that the learner-content interaction is one of three interaction components that exist between the learner and the course; the others are teacher-learner and learner-learner interactions. Hillman, Willis, and Gunawardena (1994) identified a fourth interaction component as the learner-interface interaction. A simple model of these interactions in a learner-centered context is provided in Figure 1 below.

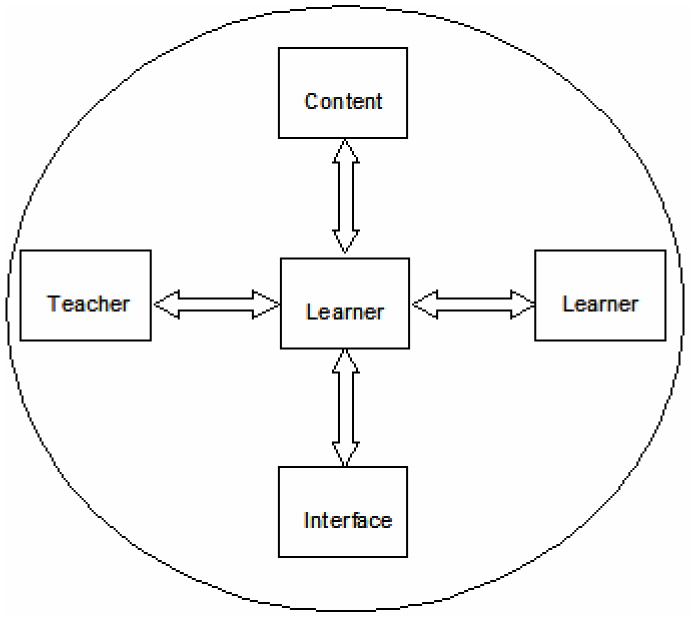


Figure 1. Course Interaction Model (based on Hillman et al., 1994; Moore, 1989)

Assuming that we can manage the learner-interface, learner-learner, and teacher-learner components through an effective course management system and a bit of faculty development effort, we look to investigate how we can effectively create the content material and make it readily available for the learner in an asynchronous environment. The immediate objective is to provide for an online course that replicates as closely as possible much of the face-to-face course while facilitating a learner-centered environment that minimizes the need for interaction between teacher and individual learner.

One such way to minimize the development effort of creating an online course is to simply record the lecture component of the face-to-face course. This can be accomplished using several different methods including videotaping with a video recorder or using a computer with a video camera to record the lecture and the content either in the face-to-face class or outside of the class. Delivering the content to the distance learner can be accomplished by mailing the recorded media in the form of videotapes or CD-ROM's, or by providing the media online.

Videotapes allow the learner to stop the lecture at any point and backup or fast-forward to review a key point in the lecture. This feature can be an important benefit in the learner-interface interaction component over a traditional lecture in that it provides the learner the independent ability to readdress lecture content that might have been unclear. However, it does not allow the student to interact directly with the instructor, an aspect of online courses that can be a significant disadvantage.

Lecture content that integrates digital video with PowerPoint slides provides additional benefits in the learner-interface interaction component over that of videotape only. These benefits include an improved user interface that allows for random access of the video content, improved navigation through the video content, and automatic synchronization between the PowerPoint slide and the related segment of video. There are several alternative products that can provide the former capability including the Tegrity™ WebLearner multimedia system (www.tegrity.com) and sofTV.Presenter (www.softv.net). Both products claim the ability to capture and integrate the lecture content consisting of voice, video, and PowerPoint slides simultaneously and allow for the on-demand access of the integrated lecture content. The sofTV.Presenter product provides an additional feature over that of the Tegrity™ WebLearner system in the user interface that allows the learner to click on a link to access the synchronized voice, video, and related Power Point slide. Providing this level of integrated lecture content is a significant improvement in the learner-interface interaction component over that of videotapes, so it is a better alternative than the use of videotaped lectures to support distance education.

Distance Education Studies

Meyer (2002) and Meyer and ERIC Clearinghouse on Higher Education (2002) provides reviews of some of the comparison studies that have been made between the outcomes of an online course and a traditional face-to-face lecture-oriented course. In particular these reviews identify several major studies which show that the use of technology in and of itself will not improve learner outcomes. Of particular interest is the research by Russell (1999).

Russell (1999) provides reviews of 355 studies on distance education from the years between 1928 and 1998. The majority of the studies reviewed are various educational applications of television and video but also include studies of multimedia, audio, radio, print, and others. Comparisons were made on test scores, grades, or other performance measure criteria of the distance education instruction with traditional face-to-face instruction. These studies found “no significant difference” between the performances of the comparison groups. This is a very promising and important finding that supports distance education instruction. It is important to note, however, that only 40 of the 355 studies specifically addressed the use of computers in the instruction. Additionally, if we consider the time frame, the majority of the studies were completed before the advent of the World Wide Web as well as effective course content management systems.

Russell (1999) contends that it is not the technology used in the course, but instead it is the redesign of the course to adapt it to the technology that will facilitate learning performance. Russell writes: "… let me reassure you that differences in outcomes [between distance education and traditional face-to-face instruction] can be made more positive by adapting the content to the technology" (p. xiii). Lynch (2002) notes that he was able to successfully replace the traditional classroom with integrated content and video recordings and that student achievement slightly improved. But most educators would agree with Clark (1994) who has argued that we can use technology as a course transmission mechanism, but it is the instructional methods that we use within any media that will improve the learning outcomes. What Russell found so frustrating was that even with all of the evidence that technology does not influence learning, people continue to believe that it does. Kozma (1994) suggests that media can influence learning if we integrate it with the appropriate teaching methods, grounding it in the "cognitive and social processes by which knowledge is constructed" rather than using a behaviorist theory.

Kozma's (1994) argument is compelling, and we should take note of it when developing a distance education course. That is, we can use technology as an aid and supporting mechanism for our distance education courses. However, in order to provide an effective distance education course, it is the effective teaching methods that we integrate with the technology we have at hand that is most important. This integration needs to also support the kinds of interactions that we typically have in the classroom. An effective distance education course is one where the learning outcomes are (at least) no worse than the face-to-face course.

What is important to understand in the context of the review by Russell (1999) is that technology is simply a tool, and without redesigning the course content, the technology tool will offer no benefit. The reliance on technology may prove to be a detriment to the learning outcomes if it is not appropriately integrated into the course. Placing lecture-oriented courses in a format that can be viewed electronically will provide a mechanism to replicate the course sufficient to provide an online version. The online version may provide the same outcomes as that of a face-to-face version assuming the learner is technologically literate and is motivated to learn in this mode. However, not all courses may be suitable for online transmission. Complexity of subject material may require the teacher to reorganize the material in a manner that provides multiple threads through the material supplemented by examples and explanations. Cognitive Flexibility Theory (CFT), a learning theory developed by Spiro, Feltovich, Jacobson, and Coulson (1992), suggests that learning complex subject material can be supported in this way.

Pilot Study

In the fall of 2003 I began the development of an online version of a course in object-oriented programming. At that time the current version of the course was typically taken by a second semester freshman student who was majoring in Computer Information Systems or Computer Information Science at an NCAA, Division II university located in the Midwest region of the United States. It was their first course in object-oriented programming. The programming language used in this course was Java. All course content materials were assembled within a course management system, WebCT. The content material included the course syllabus, presentation materials, homework, and programming assignments. These items were collected from a historical archive of the course material that had been developed and used over previous semesters in the face-to-face course. The face-to-face lecture was the primary course-content component lacking for the distance education student. It was also necessary to provide alternative support for the learner-teacher interactions that are normally provided in face-to-face mode. These included support of question and answers, office hours, and lab support. These requirements were supported by WebCT through email, discussion lists, and chat.

In order to minimize the development effort, I created recordings of the lecture material and made them available through WebCT. These lecture recordings were developed in order to provide the lecture content to the distance education students. The product I chose to create the lectures was sofTV (www.softv.net). This tool provided the capability to digitally record a PowerPoint presentation along with audio and video of the lecture. The resulting output from the sofTV product included the recorded video integrated with the PowerPoint slides with a user interface that allowed the learner to click on a slide and take them directly to that point in the lecture. I selected this tool primarily because it was the only one available without purchasing additional software and equipment, and also it pretty much did the job required. Each lecture was recorded at approximately the same time as the lectures were given during the semester. Because the material was fresh in my mind, the amount of preparation time for the recording was reduced as well. This improved the quality of the recorded lecture because I was able to better predict and address the questions that might normally be asked during the lecture.

At the end of the semester the lecture recordings were placed within WebCT and tested for functionality. In the testing process I asked several students to connect to WebCT and attempt to view one of the lecture recordings. In doing this testing, the students found that they were unable to view the recordings. Subsequently I removed the recordings from WebCT and placed them on an unsecured web site linking them through a web page. I then provided the web site to the students and asked them to retest the videos for functionality. This time the testing was successful. However, I was uncomfortable with providing the lectures on an unsecured web site. Therefore, I decided to compress each of the lectures and place within WebCT. This required the students to download the compressed lectures to their computer and uncompress them before viewing. Once again I asked several students to test this, and they were successful. Based upon these findings, my final decision was to use this compressed lecture format, storing them within WebCT, and requiring the students to download and uncompress the lectures for viewing. This completed the development effort of the course content.

In the spring of 2004 I conducted a pilot study on the learning outcomes of students in the online object-oriented programming course versus the face-to-face object-oriented programming course. In total there were two sections of the course, one, a daytime section, being the traditional face-to-face lecture version (F2F), and the other, an evening section, having lectures available online for learners to view at anytime (OL). Both sections were provided the same content material through the WebCT course management system. In effect, both sections had significant online support through WebCT, and I was the instructor for both sections. The primary difference was the lecture component.   The pre-recorded lectures were provided online to all of the OL students at the beginning of the semester. Students who had enrolled in the evening section of the course were assigned to the OL section. These students did not know ahead of time that they were going to be taking the class online. However, all students in this group consented to piloting the online course. The OL section of the course was provided with an open lab opportunity one night per week in a face-to-face setting. During the open labs, students could ask questions about their homework projects or any of the recorded lecture material. Additionally asynchronous interaction capability via email and discussion threads was provided to both the OL and the F2F sections. OL students were required to come to campus for the tests. The idea was to hold as many variables as possible constant between the two sections with the key independent variable being the presence of the instructor in video or in person.

Other variables did play a role in the differences between the two sections. These included the variables typically related to the learner-teacher interactions that are external to the classroom but were somewhat different for the OL group versus the F2F group. The reason for the differences was due to the time and location differences between the OL group and the instructor. Because the OL students did not have access to me during my face-to-face office hours, general questions, office hours, and lab support for the OL student was facilitated by WebCT through email and discussion lists. Both the OL and F2F groups could participate in the discussion lists and could post direct questions to me by email, however the F2F group had additional access to me during my face-to-face office hours.

Technical requirements for the OL group were different from those in the F2F group. For the F2F group all technical requirements were supported in the classroom as well as several other alternative lab locations. These facilities were also available to the OL group as well. However, in order to function in the OL group without driving to campus, the OL person needed to have the following basic technical platform capabilities: a relatively fast computer (for example a 400 Megahertz processor) with a minimum of 128MB of RAM, a high speed internet connection, and a Java compiler and runtime environment which was freely available.

The data analysis plan for this pilot study was to compare the test scores and the combined homework and programming assignment scores between each of the sections with the hope that there would be no significant difference between them. Three separate planned pairwise comparisons were made using a one-way between-subjects ANOVA design for each. There were no pretests given. Two course prerequisites were listed in the course catalogue and syllabus which provided some assurance that the level of students remained consistent between the two sections. Results revealed a significant difference in achievement on the exams favoring the traditional course, but no significant difference in overall homework scores (see Table 1 and Table 2).

Because there was a significant difference, further analysis was performed on the completion of the prerequisites by the students in order to determine if there was an initial difference between the two groups. This analysis showed that in fact there were several students who had not completed the prerequisite courses in the OL section. Also, because of the small sample sizes and the differences in background education between the day students and the evening students the findings are limited. However, because there was a difference between the two groups, the decision was made to provide face-to-face lectures to the distance education students for the remaining portion of the course.

Table 1. Descriptive Pilot Study Statistics for Online (OL) and Face-to-Face (F2F) Groups

Source	**Mean**	***N***	***SD***
T1 F2F	81.69	13	10.1
T1 OL	70.33	6	3.27
T2 F2F	80.85	13	11.42
T2 OL	68.00	6	11.68
HW F2F	83.46	13	13.56
HW OL	78.00	6	14.81

Notes. T1, T2, and HW refer to test 1, test 2, and homework scores. OL refers to online group and F2F refers to face-to-face group.

Table 2. ANOVA Pilot Study Statistics for Online (OL) vs. Face-to-Face (F2F) Groups

Variable	**Source**	***df***	***SS***	***MS***	***F***	***p***
T1	Between Groups	1	529.69	529.69	7.05	0.017*
	Within Groups	17	1278.10	75.18		
T2	Between Groups	1	677.47	677.47	5.12	0.037*
	Within Groups	17	2247.69	132.22		
HW	Between Groups	1	122.45	122.45	0.63	0.438
	Within Groups	17	3303.23	194.31		

Note. T1, T2, and HW refer to test 1, test 2, and homework scores.

*$p < .05$

Throughout the pilot study group interviews were held with the OL section of the students during the open lab. It was my intent in these discussions to gain an understanding of some of the problems in the OL course. These discussions had no structured format and typically only about three of the OL students were represented. However some interesting issues were raised. These included the following:

- It was more difficult to learn the material without having the teacher available for immediate interaction.
- Students did not always take advantage of viewing the video. However, the face-to-face students did not always attend class either. These data were anecdotally collected based upon my perceptions of the types of questions that the online students asked as well as the number of unoccupied sets in the F2F section. No empirical data on attendance for either of the sections were recorded.

- More examples of working programs would have been a big benefit to the OL students.
- The video lectures were easy enough to use and had adequate quality.
- Students really enjoyed having the flexibility to view the lectures at anytime.
- Only two or three students took advantage on a regular basis of the open lab opportunities provided for OL section.
- Abstract concepts were difficult to grasp without having the direct feedback.
- Regular homework assignments along with the programming assignments helped them to focus on the important issues.
- Students preferred to see programs demonstrated and discussed as part of the demonstration. Although a walkthrough of some of the programs was presented in the videos, an actual demonstration was not available to the OL group. Working versions of the programs were available, however, to both the OL and F2F groups to download and run on their computers. A detailed discussion of these programs could be provided in the future as teaching cases through a web interface as an instructional supplement for both groups.
- Some students preferred to learn by doing their programs in class. The students liked the feedback that they received immediately from me while they worked on their program. This was not available to the OL group unless they drove to campus for the open labs or made a face-to-face meeting with me.
- Downloading the videos with a slow internet connection was a problem.

Difficulties in Teaching Object-Oriented Programming

Raab, Rasala, and Proulx (2000) suggest that the cross-platform capabilities of Java and the robust graphical user interface (GUI) components provide a great argument for using Java to teach programming. But the complexity of building a complicated GUI in a CS1 course (first programming course in computer science) is problematic. Raab et al. suggest using a toolkit of pre-developed classes that can be used as the framework for beginners to build from. This allows them to solve more complicated problems that are much more interesting and that illustrate the breadth of computer science.

Yang & Wei (1999) used a project-based approach to teaching C++, an alternative object-oriented language to Java. In their study they connected individual assignments into a semester-long project. Using this strategy, they felt that the important topics such as encapsulation, code reuse, and software design were repeatedly addressed throughout the course. However, their findings were not very conclusive with regard to using a semester-long project in a project-based approach to teaching C++. Interpretation of the student responses suggested most students did not seem to mind working on the same problem throughout the semester although the survey was open ended and did not ask this question specifically. Some students also commented that the assignments were too difficult. Tutors for the course, who were students who had completed the course the previous semester when it was not project-based, also felt that the assignments were more difficult than the previous semester. Most interesting, though, was the finding that the mean course grade was down from an average of 3.4 on a 4.0 scale to 2.8, *n* = 57, and in the following course was down from 3.2 to 2.8, *n* = 23. One of the reasons suggested for the drop in the mean course grade was the fact that there are an increasing number of non-majors who were taking the course. This finding possibly suggested the need for a change in the teaching methods to be more sensitive to the requirements of students who bring a different technical background to the course. Because the research methods used by Yang and Wei (1999) are somewhat confusing it is difficult to determine if using a project-based approach to teaching an object-oriented language is a better method of instruction than one that does not use a project-based approach.

A major difficulty with teaching Java is that difficult concepts must be addressed at an early stage (Biddle and Tempero, 1998). Although it has become a popular language, it may be only a marginally better teaching language than C++. Even the writing of a simple one-line program in Java requires the introduction of complex concepts such as inheritance, static methods, or exceptions. Because these concepts are not easily addressed for beginners, some universities continue to use a traditional structured programming language such as C in their first programming course. This choice, however, can be problematic as well. Students often report that because object-oriented programming is so much different than structured programming, it is difficult to unlearn some of the bad habits that are learned with a language such as C when making the switch to Java.

Visual Basic is an object-based programming language in which difficult concepts such as inheritance and polymorphism are not available to the programmer. In a study by Madden and Chambers (2002) regarding student attitudes towards learning Java, over 50% of the students reported that learning Java was easier than learning C, and for C++ the findings were similar. One of the reported reasons in their study that learning Java was easier than learning C++ is that the Java syntax is simpler to learn and understand. As would be expected, students reported that prior knowledge of C made the learning of Java easier (8% disagreed), and students reported that knowledge of C++ made the learning of Java easier (24% disagreed). However, only a minority of the students reported that prior experience in Visual Basic made a difference in learning Java. It can be inferred from this study that learning Java was really no different than learning Visual Basic when using an object-based programming approach for Java. Related to this finding is that object-based programming was not considered to be difficult whereas object-oriented programming concepts were more difficult to learn. The primary difference between object-oriented and object-based programming in Java is that object-based programming utilizes a reduced set of capabilities or features of the programming language. Object-oriented features of polymorphism and inheritance are not used when doing object-based programming. Perhaps treating Java **initially** as an object-based programming language may make it simpler to learn **initially**. However, even simple programs in Java introduce complex object-oriented concepts (Biddle and Tempero, 1998), and it is very difficult to avoid discussion of these concepts.

Integrated Development Environments (IDEs) also play a significant role in learning Java. Many of the early IDEs for Java were very difficult to use. Because of this difficulty, quite often a command-line interface is used with no support for interactive software debugging capabilities. In fact Kolling (1999) reported that educators found the lack of an adequate IDE was the biggest problem in teaching Java. Evolution of the requirements for a suitable IDE include ease of use, integrated tools such as a source code editor, compiler, and debugger, support for code reuse, student learning support allowing for experimentation and interaction, support for group work for teams of programmers, and availability which includes being reasonably priced and easy to install (Kolling, 1999). The tool BlueJ was developed to address these needs (Kolling, Quig, and Patterson, 2003). BlueJ is a software development environment that allows the learner to interactively create objects in an environment that hides many of the complexities of Java. It is an environment which encourages experimentation and interaction to improve the learning of the necessary complexities of object-oriented programming (Kolling, 2004) but without the unnecessary complexities.

The complexity and instability of teaching an introductory computer science course is also documented by Roberts (2004). Roberts notes that the number of programming details that a student must master has grown much faster than the corresponding number of high-level concepts (complexity). For example some of the programming details which Roberts is referring to include additional language constructs as well as the number of supporting standard code libraries that are required to write a program. High-level concepts such as writing functions that serve a single purpose have remained relatively the same. He goes on to say that the languages, libraries, and tools on which introductory computer science education depends upon are changing more rapidly than in the past (instability). Roberts suggests that both economic and technical forces encourage these factors however there is no *a priori* reason to conclude that it is impossible to teach the essential object-oriented programming concepts in a simple and stable environment. The essential complexities of Java include encapsulation, inheritance, polymorphism, reuse, etc., whereas the unnecessary complexities include the magnitude of the Java 2 class libraries (some 50,000 library functions) and the rapid obsolescence of libraries and tools that are available for Java. Libraries and tools become obsolete because Java is still a relatively new language and is still evolving. Unnecessary complexities are also illustrated by the differences in the size of the textbooks that are now used to teach Java. One of the more popular books (Deitel and Deitel, 2003) has 1536 pages of text whereas the classic *Pascal User Manual and Report* (Jensen and Wirth, 1991) that was used to teach Pascal had about 226 pages.

Scaffolding Student Understanding with CBR

At this point it should be clear that learning object-oriented programming, and in particular, Java, is a difficult task. Java is complex, requiring the understanding of abstract concepts, technical complexities, and a massive set of software libraries. Threading important topics into a semester-long project might be a way to improve students' understanding. However, not all students have similar backgrounds and because of their dissimilar backgrounds a single project might not work. Difficult concepts are necessarily addressed at an early stage or they are confounded due to previous background knowledge with a structured programming language. Treating Java as an object-based programming language does limit some of these concepts and can benefit understanding. Additionally, providing effective tools that allow the student to simulate the creation and manipulation of objects in an interactive environment is also a benefit. In the end, learners should be provided with the scaffolding to support their understanding. One alternative is to provide this support through case-based reasoning.

Case-based reasoning (CBR) is a learning model that incorporates problem solving, understanding, and learning, integrating them with memory processes. CBR supports access to prior cases for both reuse and adaptation. Both new and adapted cases can be stored for future use, thus the learning occurs as a natural consequence of reasoning (Kolodner, 1993). In more simple terms CBR is a learning model that suggests that people learn from their past experiences, and they use remindings of these past experiences to solve new problems. Providing the student with a tool for learning Java that is based on the CBR model should provide the scaffolding to support the learning of the difficult concepts encountered with the Java programming language. Examples can be used to demonstrate cohesion between the concepts and features of the language and real-life problems. Students will learn by example as well as adapt new examples. One approach to providing this type of system is through a web-based hypertext environment.

Implications of Background Research

From the background research the researcher found that distance education courses could be developed that are at least as effective as face-to-face courses. Improvements to video technology suggest improved methods of lecture content transmission that should be considered and tested. At the same time, some particular courses may not be viable candidates for online courses. The pilot-study raised this as an issue. Some courses, such as one in object-oriented programming, contain difficult concepts that may require additional scaffolding for the learner to gain sufficient understanding. Email and discussion lists may not be sufficient. Therefore we considered in this study the effects of a case-based reasoning system as a support for the complex concepts. These were the issues that this research attempted to address.

Significance

The primary benefit to the completed study is that it suggests the appropriate use of a case-based reasoning tool to enhance our ability to teach introductory programming classes whether or not they are online. The study also tested the applicability of the case-based reasoning model to the learning of an object-oriented programming language. Additionally, the study performed a correlational analysis of the relationships of motivation, background knowledge, and student attitudes with learner competence. Universities recognize the new audience that an online course might service however they also recognize the importance of quality instruction. Thus, this study also suggests new sets of courses that might be offered online with the support of a case-based reasoning tool that have traditionally not been offered in the past because of differences in quality.

The study should be of particular interest to college level instructors who provide instruction in a programming language. The instructional content and strategy developed in support of this study is flexible enough to support transmission of any programming language instruction.

Research Questions

Below is a list of the research questions that were addressed by this study. Learner competencies were identified as knowledge, comprehension, application, analysis, synthesis, and evaluation.

1. Is there a statistically significant difference in the performance on simple questions between the Case-Based Reasoning Assisted (CBA) group and the Lecture Notes Only (LNO) group?
2. Is there a statistically significant difference in the performance on complex questions between the CBA group and the LNO group?
3. How is student performance on the measures of learner competencies related to motivation, background knowledge, and attitudes towards the case-based hypertext learning tool?

Limitations of the Study

It was the goal of this study to do a quantitative analysis with as much statistical rigor as possible. The subjects used for this study were randomly assigned to one of two groups. However, the pool of subjects was limited to those students who had enrolled in Object-Oriented Programming 1 at an NCAA, Division II university located in the Midwest region of the United States. Therefore, it is more difficult to generalize this study to a much broader population, and that limits its external validity.

The number of subjects participating in the study was rather small; I could not anticipate or affect the class sizes. For example, the number of students in the online group of the pilot study was initially eight. In order to have sufficient power in the study, it is important to have a large enough sample size (Keppel, Saufley, and Tokunaga, 2003 p. 209). A rough estimate of the sample size suggests that there should be at least 20 subjects in each group.

The ordering of the two types of instructional support is also a limitation of the study. The case-based hypertext tool was not used as an instructional aid until midway through the course. It could be argued that by that time the students do not require any additional support. They may have learned how to use the existing resources to support their learning. Thus, there may be no significant difference between the performance of the students with or without the case-based hypertext tool as an instructional aid. However, it is only at about the midpoint of the course where the concepts become more complex and ill-structured. So, introducing the case-based hypertext tool at that time was appropriate. Other sequencing situations arise as well, but because of the number of groups, the limitations of the sample sizes, and the ethical requirement to provide all students the same aids, this was the best that could be done.

Other extraneous variables may have affected the outcome of the study. In particular it was difficult to determine how much of the content within the case-based hypertext tool was actually read by each subject. The only guarantee that a student accessed the tool was that they submitted the example. However, there was incentive for them to read the content in that it aided them in creating an example (which was required), and it helped them prepare for the graded posttest, therefore the impact to the results of the study were minimal. Finally, not all of the students submitted examples in a timely basis. In order to ensure that all of the subjects submitted an example, several directed emails were sent. No special coaching on creating the example occurred, therefore impact to the results of the study were also minimal.

Some of the limitations to the external validity were eliminated by limiting the differences in the treatment groups to one particular variable which was the type of instruction support. For the duration of the study the transmission of the course to all subjects was the same, that is, the transmission was online. One can argue that the viewing of a digital video can be done anytime and as such is another variable in the experiment. However, for the purposes of this study the time and space dependencies were subsumed in the instructional mode.

Definition of Terms

Case-Based Reasoning. Abbreviated as *CBR*, a problem-solving paradigm that utilizes the specific knowledge of previous experiences and concrete problems in order to solve new or similar situations (Aamodt and Plaza, 1994).

Cognitive Flexibility Theory. Abbreviated as *CFT*, a constructivist theory of learning that suggests advanced learning in ill-structured domains must be supported through alternative cases and multiple paths through a set of knowledge content (Spiro et al., 1992).

Command-line interface. A method of interacting with the computer in a text-based mode. "The user sees the command line on the monitor and a prompt that is waiting to accept instructions from the user. The user types in the command, the computer acts on that command and then issues a new prompt for the next instruction from the user" (Webopedia, 2004).

Distance education. Structured learning in which the student and the instructor are separated by time and place (McIsaac and Gunawardena, 1996).

Hypertext. **“**Non-sequential writing—text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways” (Nelson, 1990). A database format in which information related to that on a display can be accessed directly from the display (Merriam-Webster Inc., 2004).

Integrated Development Environment. Abbreviated as *IDE*, a programming environment integrated into a software application that provides a graphical user interface builder, a text or code editor, a compiler and/or interpreter and a debugger. Visual Studio, Delphi, JBuilder, FrontPage and DreamWeaver are all examples of *IDE*s (Webopedia, 2004).

Object. A self-contained software module that includes both the data and the procedures (the programming commands) which operate on that data (Williams and Cummings, 1993).

Object-based programming. “A method of programming in which programs are organized as cooperative collections of objects, each of which represents an instance of some type, and whose types are all members of a hierarchy of types united via other than inheritance relationships” (Spencer, 1993).

Object-oriented programming. A method of programming in which programs are put together from self-contained software modules called *objects.* Object-oriented programming makes it easier to generate new objects that automatically “inherit” the capabilities of existing objects. The programmer can then extend the inherited functionality in the new object (Williams and Cummings, 1993) .

Online. Connected to, served by, or available through a system and especially a computer or telecommunications system (as the Internet) (Merriam-Webster Inc., 2004).

Online course. A course which uses Web-based or Internet-based technologies in order to provide course content and interaction capabilities. In the context of this paper *online course* is one form of a distance education as well.

Semi-automated reasoning tool. A software-based learning tool that is primarily implemented through hypertext links that provide static content through a question and answer interface. The automated component allows the student to add examples and index them appropriately. Human review by an expert is required to ensure the correctness of the example.

Chapter II: Review of the Literature

Introduction

The purpose of this research was two-fold. The first purpose was to determine whether a student can perform as well in an online course in object-oriented programming as the student in a face-to-face object-oriented programming course. The second purpose was to determine if a CBR tool will improve a student's understanding of the more difficult concepts in an object-oriented programming course. Related research on the use of videos in distance education as well as applications of cognitive-flexibility theory towards understanding existed. However, there was currently no public record of research on the combination of these variables and in particular how they might be used to support understanding in an object-oriented programming course in Java.

The purpose of this literature review is to address the instructional design techniques and theory that could be incorporated into effective course content supporting an object-oriented programming course in Java. Although Java has a well defined syntax and structure, the concepts that it supports such as objects, inheritance, and polymorphism are ill-structured and complex. Therefore, the literature review will be developed as follows: The first section describes case-based reasoning and its close relationship with a constructivist theory of learning, Cognitive Flexibility Theory (*CFT*) (Spiro et al., 1992), which supports learning in ill-structured and complex domains. The second section will present a set of teaching architectures and their impacts on instructional design. The third section explores the use of hypertext videos to support distance education. The fourth section reviews alternative case-based reasoning implementations supporting CFT and their viability in an online instructional setting. Finally, a summary of the literature review is provided along with its implications for this study.

Case-Based Reasoning

Case-based reasoning (CBR) is a problem-solving paradigm that utilizes the specific knowledge of previous experiences and concrete problems in order to solve new or similar situations (Aamodt and Plaza, 1994). CBR is a constructivist learning theory which suggests that we build our knowledge from our previous experiences. Bruner's (1996) constructivist tenet in education shares similar features to that of knowledge building. Bruner writes that "reality construction is the product of meaning making shaped by a culture's toolkit of ways of thought" (p. 19). Bruner goes on to say that education should be "…helping people become better architects and better builders" (p. 20). Much of the early research in CBR was based upon how remindings are used in human reasoning (Schank, 1982).

CBR is a constructivist learning theory that supports incremental learning. Learning is incremental when new knowledge is stored for future use when it is learned. In CBR both our failures and our successes in solving new problems are cataloged and stored for future use, thus building knowledge incrementally. CBR can be used to guide the design of a computerized learning environment. It is within this context that I will be referring to CBR unless the context is otherwise explicitly stated.

A CBR-grounded learning environment provides the learner with concrete examples, i.e. actual cases, rather than abstract rules. Experience is provided by means of a case library that has cause, effect, and lessons learned components. Learners may access these cases through multiple indexes. Incorporating an appropriate set of indexes over these experiences provides the learner with alternative views into the same sets of cases.

Providing a learner with multiple ways to traverse the content will not necessarily lead to improved learning. In fact it may even lead to a "confusing labyrinth of incidental or ad hoc connections" (Spiro et al., 1992, p. 68). Instead the content should be organized in way that a “learner sees a range of conceptual applications close together, so that conceptual variability can be easily examined" (Spiro et al., 1992, p. 68).

CBR in Context

Leake (1996) identifies two tenets of nature which underlie the CBR paradigm:

1. Similar problems have similar solutions.
2. The types of problems encountered by a learner tend to recur.

Thus, the primary source of knowledge is a set of cases in memory which can adapted to new situations. A learner, whether it is machine or human, builds upon knowledge of both successes and failures. A CBR system is a machine system that explicitly integrates memory, learning, and reasoning (Kolodner and Guzdial, 2000). As the knowledge base grows, the CBR system’s ability to solve both new and similar situations grows as well, much like what occurs in the human process of natural learning. In computer science when machine learning occurs only when new problems are encountered and solved, the learning is known as lazy learning (Mitchell, 1997). Kolodner and Guzdial (2000) point out that a CBR system can (a) suggest resources through well indexed case libraries and lessons learned, (b) suggest activities through the writing of new cases, (c) suggest ways of moving the learning forward through sharing of the cases, and (d) suggest ways of creating useful case libraries by seeding the library with an initial set of cases and then letting it grow and evolve. A case library grows based upon the new situations that are solved and then stored in the case library.

Examples of CBR

From a cognitive context the CBR learning theory is often applied routinely in a natural form of problem solving. Often we are reminded of similar situations and we use them to solve a new situation at hand. The following examples illustrate where CBR might be useful and how it might support problem solving.

1. A teacher is trying to determine if she is breaking any copyright laws when she purchased a single copy of a workbook and wants to make copies of some of the pages for her students. She looks for similar situations and how they were approved or not approved by the authority at her school. Based upon these remindings she makes the decision not to copy the pages.

   A CBR system could support this problem by providing an index such as a categorized list of similar situations related to making copies, copyright issues, and how they were handled. Similar situations where copyright is involved might suggest related school policies that aid the teacher's decision making. Based upon the information provided, the teacher can make an informed decision. Additionally the teacher can log the situation and its resolution for future use.

2. A speech pathologist is trying to evaluate a child's speech disorder that occurs only in high stress situations. He is reminded of several other children who had a similar disorder and reviews how they were treated and the success or the failure of the treatment. Based upon these remindings the speech pathologist recommends a treatment. The treatment, its success, or its failure is recorded for future reference.

   A CBR system could support this application by providing an index to a set of related cases based upon the diagnosed disorder. It could suggest treatments as well as provide narratives of what to expect and how well the treatment worked. Alternatively the treatment could be indexed as well. Through this index the CBR system could provide a list of all of the disorders that the treatment has been used for in the past. This list of disorders might suggest an alternative diagnosis to the speech pathologist.

Types of CBR

CBR is, in essence, a paradigm covering a set of alternative methods that are based on accessing and using remindings. A reminding is nothing more than a past experience whose memory is elicited when a similar situation arises. This experience could be a previous conversation, cause-effect remembrance, or a situation in which the learner found herself before. These remindings are stored in a database of cases which can be accessed by the human learner as well as the computer in order to solve a new or similar problem. The following is a list of the some of the primary types of CBR systems from Aamodt and Plaza (1994) including a brief discussion of each.

Exemplar-Based Reasoning. Exemplar-based reasoning (EBR) employs nothing more than classification of a new case based on how previous cases were classified. For example suppose a human learner was trying to determine how to rate a movie G, PG, PG-13, or R. Using the case library and an EBR system the learner could search for similar movies with similar attributes and see how they were classified. Based upon how these were classified, the human learner could then classify the new movie similarly. Because not all of the similar movies were perhaps classified the same, i.e. some may have been G and some PG, the human learner could look at how the majority of similar movies were classified and use that as his or her classification scheme. The human learner has the final say in the classification process and thus can override the suggested classification that was made by the CBR system. The movie and its final classification are stored in the case library.

A close relative of EBR is instance-based reasoning. Instance-based reasoning systems operate much the same as exemplar-based reasoning systems except that the computer system does not allow for the human learner to override the solution.

Memory-Based Reasoning. In memory-based reasoning (MBR) decisions are based upon memories of specific events versus making decisions based upon relationships or rules built up from experience (Stanfill and Waltz, 1988). For example, we remember a situation but we have no knowledge of how we got there or its relationship to anything else. An MBR system looks for those cases that match on indexes that are as close to the current situation as possible. The remindings are represented by feature vectors (i.e., records) linking similar remindings through matching feature (i.e., field) values. An MBR system's search function works directly on the data looking for syntactic patterns in the database (Stanfill and Waltz, 1986). Thus, computational processes that provide the human learner the most relevant cases based upon a matching of the index attributes are the ones employed in this method. Providing the cases in a relevance order like the output of a search engine allows the learner to explore the related cases to determine if they fit. How best to organize the memory separates MBR from other CBR methods.

Analogy-Based Reasoning. In analogy-based reasoning (ABR) the human learner solves the new situations with past experiences from a different domain whereas in CBR the past experiences are from the same domain. For example, in solving new computing applications we often look to biology and human cognition to help create the new solutions. In education we often provide analogies between disciplines in order to support learner understanding. Providing the human learner with these cross-domain analogies in an ABR computer program is a quite complex task, but it offers a tremendous area of growth in the field of the CBR paradigm.

Case-Based Reasoning. Although the term CBR (as used to mean a process) is often confused with the actual CBR paradigm (as used to mean a theory about human and machine learning that involves using this process), it does have a specific meaning as well. CBR distinguishes itself in that it employs more than a simple index structure to the case-base. The cases are more complex than those processed using MBR or EBR. Cases are linked through a rich network of indexes, relationships, and structures and the cases are also supported by background knowledge. In a computer implementation this background knowledge could be a set of if-then rules or decision trees. Additionally, not all cases in the case library are complete but may be sub-cases that can be rolled together to solve the problem.

Mitchell (1997) summarizes the characteristics of the CBR process as an instance-based method in which the cases may be rich relational descriptions, the retrieval and recombination process relies on both the case library and general knowledge, and there tends to be a tight coupling in the computer implementation among the retrieval, the reasoning from the general knowledge, and the problem solving. Mitchell's summarization implies that the cases are often narrative in nature. The cases have the ability to be accessed through various sets of indexes which are conceptually linked. And, CBR is a process based upon the CBR paradigm which supports knowledge building by using prior experience and generalized knowledge. Thus, the CBR process as implemented in a computer application is both grounded in constructivist learning theory and also it has the capability to support a human learner in constructivist learning.

The CBR Learning Cycle and its Implementation

The CBR Learning Cycle depicted in Figure 2 can be generalized to have the following four steps (Aamodt and Plaza, 1994):

- Retrieve the most similar case or cases
- Reuse the information and knowledge in that case to solve the problem
- Revise the proposed solution
- Retain the parts of the experience likely to be useful for future problem solving

Although Aamodt and Plaza (1994) are speaking in computational terms, this process is very analogous to a human's cognitive process for problem solving. Remindings are retrieved to solve new problems. These remindings could cross domains and could be used in an analogy process. These retrieved experiences are then reused to solve the new problem. The solved problem may include a revision to an existing case or a combination of solutions from more than one case. The solution is then tested by the learner and some level of success or failure is recorded. Based upon this feedback, the case and its solution are tagged appropriately for further reference. This process is additionally supported by the learner's general knowledge of the problem domain.

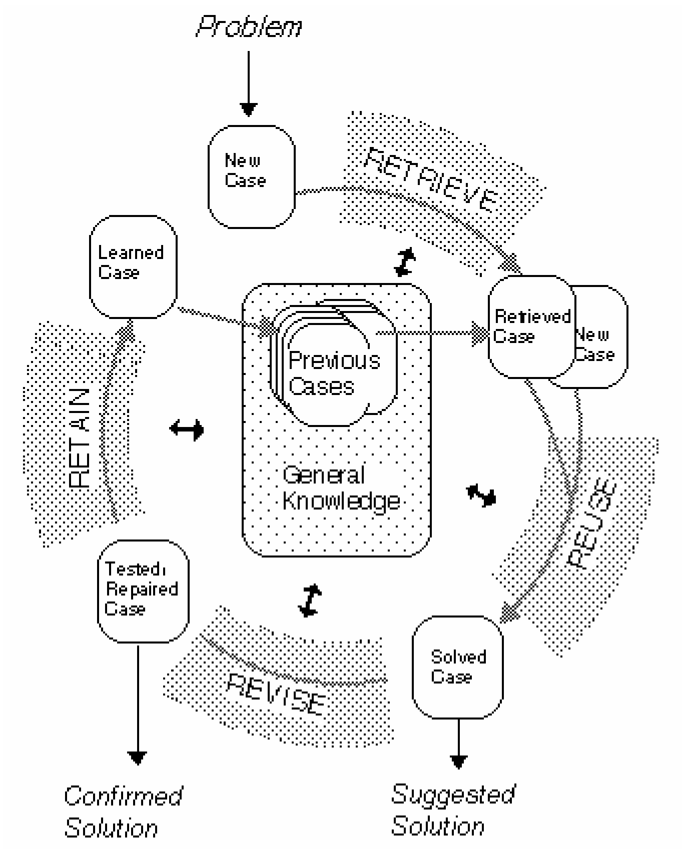


Figure 2. The CBR Learning Cycle (reprinted with permission from Aamodt & Plaza, 1994)

Computer implementation of the CBR Learning Cycle requires that a knowledge acquisition and engineering process take place first. This includes defining the cases electronically and providing the proper index and search capability. It often requires that experts be interviewed and their solutions to problems be recorded. Additionally, background knowledge is implemented in some fashion such as if-then rules, decision trees, or any number of mechanisms. An interface is developed that allows a learner to query the case-base and retrieve cases that are similar to the one that needs to be solved. If similar cases are not found, then others may be generated that are either derived or are a combination of more than one case. This intelligent system may need to access the knowledge base to assist in the derivation. The learner has the option to create whole new cases if necessary. Finally, the computer implementation must provide the mechanism to remember the new situation and how it was eventually solved. This may include both success and failure information.

CBR and Cognitive Flexibility Theory

Spiro et al. (1992) describe a constructivist theory of learning, Cognitive Flexibility Theory (CFT), which suggests that advanced learning in ill-structured domains must be supported through alternative cases and multiple, criss-crossing paths through a set of knowledge content. They suggest that the complexity of these types of domains cannot simply be understood in a single pass. Learning environments supporting complex and ill-structured domains should be formulated to address factors contributing to the failure to learn complex knowledge at advanced instructional levels (Spiro et al., 1992). The learning environment must have the ability to present multiple cases or representations of the same concept in a variety of different ways in order to illustrate the concept's complexity and ill-structuredness. Their argument is that simply providing a set of situations and their solutions in order to illustrate a concept is not sufficient to provide support for advanced learning. Instead, the learner must have the ability to construct solutions to new situations from the knowledge content that includes a case library supported by multiple indexes and generalized knowledge where appropriate. This ability to adapt and form new knowledge content can be implemented using theory-based hypertext systems which incorporate this flexibility. One of the theory-based alternatives is a hypertext system with a built-in case-based reasoner grounded by CFT.

CFT is actually a theory of case-based learning. A case-based learning environment should incorporate features that support the five principles of CFT as identified by Spiro and Jacobson (1995). These include:

- Use multiple conceptual representations of knowledge
- Link and tailor abstract concepts to different case examples
- Introduce domain complexity early
- Stress interrelated and web-like nature of knowledge
- Encourage knowledge assembly

By the very nature of a system built around case-based reasoning, these principles are upheld. For example, use of analogies to identify similar cases is an example of using multiple conceptual representations of knowledge. These analogies can be set up to counteract the specific negative effects of each other in order to demonstrate a concept's complexity and ill-structuredness (Spiro, Coulson, Feltovich, & Anderson, 1988). An alternative to this might be to use alternate points of view, for example student, instructor, or administrator, as indexes into a case library. Again this supports the multiple conceptual representation of knowledge. CBR systems provide real-life case examples as opposed to abstract concepts. This technique supports the second principle of linking and tailoring abstract concepts to different case examples. CBR systems provide cases that entail relationships that are not learned in isolation, but rather packaged together. This feature supports the third principle that domain complexity should be introduced early. With multiple indexes into its case library, a CBR system provides a web-like representation of knowledge stressing interrelationships between the cases. This mechanism supports the fourth principle. Finally, a CBR system provides multiple alternative cases to solve the same problem. In the event that no good solution exists, a CBR system can provide components that might be used to solve the problem with suggestions about how these components could be combined. Thus knowledge assembly is encouraged which is the fifth principle.

Instructional Design and Teaching Architectures

A typical classroom approach in education is to stand at the front of the class and spoon-feed facts to students in various ways. In this approach we assess the knowledge of the student through testing against these facts. We, as educators, are encouraged to get all of the students to the same place at the same time. As suggested by Schank and Cleary (1995), there is very little that is natural about this process. In Schank and Cleary's learning theory they suggest that people learn by doing. They build knowledge through reasoning based upon what they remember or their past experiences. More specifically, they use case-based reasoning to solve new problems. Case-based reasoning in a nutshell implies that learners solve problems when being reminded of previous experiences that they have been faced with. These previous experiences may be simply in the form of stories with no particular structure (Schank, 1990). Case-based reasoning engages the learner in active problem solving encouraging and supporting both an active and creative learner to construct his or her knowledge from the content (Perkins, 1999). This approach is based on constructivist theory and is significantly different from a response to a stimulus approach based on a behaviorist theory of learning (Wilson and Myers, 2000).

Using past experiences is a thoughtful and reflective process. It follows the participation metaphor as described by Sfard (1998). By developing schematic generalizations over time based upon these experiences, a learner has a much better chance of retaining the newly acquired knowledge. Alternatively, rules or generalizations that are memorized will often be quickly forgotten as they have not been instantiated by a set of cases. From the tenet that people learn and gain knowledge through their past experiences, Schank and Cleary (1995) laid out a set of what they call "teaching architectures." These teaching architectures include case-based teaching, natural learning, learning by doing, incidental learning, learning by reflection, and learning by exploring. Each architectural component is described and summarized below as well as linked to foundational research. From these architectures we build the foundation for the appropriateness of hypertext videos and the design and use of a case-based reasoning system to support an object-oriented programming course in Java.

Case-Based Teaching

People learn from their successes and their failures. We often reason from similar cases and draw conclusions from how they were handled both in their success and their failures. This is often how we learn. Experts reason with whole libraries of cases to perform functions such as medical diagnosis (Bruno, 2000) or weather prediction. Providing examples for learners to draw from and reason from is routinely used in textbooks. For example, in learning math we are typically presented real world problems with detailed solutions.

Students are natural case-based reasoners, and Schank and Cleary (1995) suggest that teachers need to play three different roles in order to leverage this ability. These include:

1. Providing students with the applicable cases at the time they are required.
2. Helping students explore and draw out useful generalizations from cases.
3. Placing students in situations in which they will face failure.

Of these three roles, learning from failure, requires a bit more explanation. Failures can be catastrophic, but typically they are not. Schank and Cleary (1995) refer to these types of non-catastrophic failures as “expectation failures.” An expectation failure is failure that occurs where you get one outcome when you are expecting another. These are the types of failures that we often learn from. For example suppose you ordered seafood at a restaurant that specializes in steak and the seafood turned out to be distasteful. You might generalize that in the future you should order an item from the restaurant that the restaurant specializes it. This is an example of learning from an expectation failure.

This whole concept of case-based teaching and reasoning is the focus of a vast amount of literature and research which demonstrates its applicability to teaching and learning. A prime example of the current endeavors in case-based teaching is the collection of research, conferences, workshops, list of practitioners, case ideas, and more that can be found at the National Center for Case Study Teaching in Science http://ublib.buffalo.edu/libraries/projects/cases/case.html.

Natural Learning

People have natural mechanisms for learning that allow them to understand a large variety of material over time. In natural learning, a person draws conclusions from their experiences and the wonderment about them. The natural learning process consists of three steps (R. C. Schank and Cleary, 1995):

1. Adopt a goal.
2. Generate a question.
3. Develop an answer.

For example some people love politics and therefore learn much about politics on their own. Others may enjoy golf as a past-time and have the goal to become a competitive golfer. They generate questions about how to improve their game. They learn through practice and consultation on how to improve their swing, thus improving their game. This provides them with an answer and makes them a better golfer.

Course interactions should be developed in such a way that they support the natural learning mechanisms that learners have. An instructional design that has a rigid framework will not support this learning mechanism. Course interactions should be structured in a manner that allows learners to develop their own goals within the course context. Based upon these goals the course interactions should allow the learners to generate questions and develop related answers. If structured correctly, the course-content can provide much of the expert knowledge that is typically available through the teacher-learner interaction. For example, by providing a guided interface over an exploratory hypermedia system (Nelson, 1994) with a large amount of alternative topics expert knowledge is provided as well as allowing the learner the capability to independently choose the topics that are important him or her.

Learning by Doing

The only way to really learn something is to actually do it. Can you really learn how to drive a car unless you actually drive the car? Can you learn how to play basketball without actually playing basketball? People learn best by doing. Sometimes people are left to learn things on their own, by themselves. Other times, they may need support from an expert such as a flight instructor.   David Kolb (1984) describes learning by doing (experiential learning) as a four-stage process (see Figure 3).

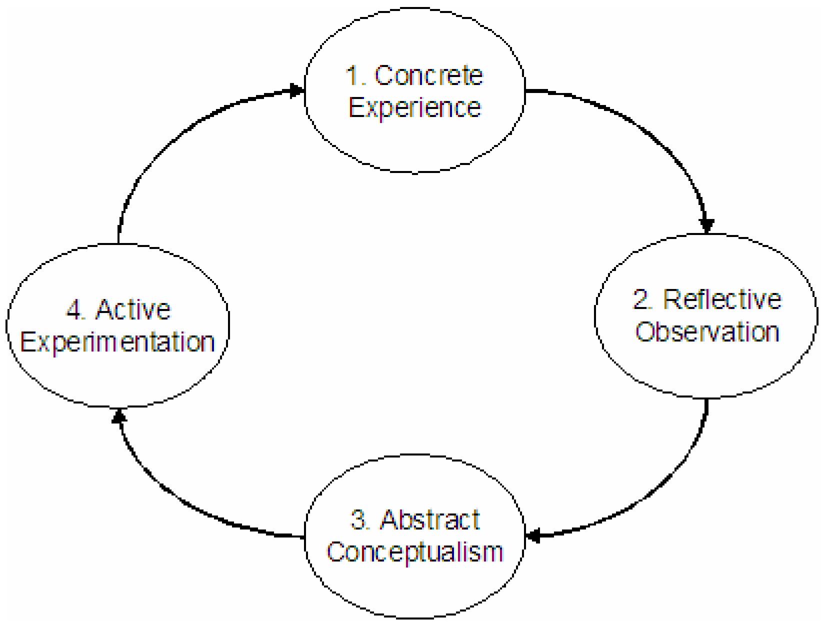

1. Concrete Experience
2. Reflective Observation
3. Abstract Conceptualism
4. Active Experimentation

Figure 3. Learning by Doing Cycle (adapted from Kolb, 1984)

The four stages are further defined as follows:

1. Concrete experience (sensing/feeling)
2. Reflective observation (watching)
3. Abstract conceptualism (thinking)
4. Active experimentation (doing)

Schank and Cleary (1995) suggest educators and schools should make use of learning by doing in their educational processes and providing built-in mentor support for the experiential learning as required. However, individualized mentor support is typically not available because of cost considerations and lack of access to an expert. Additionally, some real-life situations cannot be created. They must be simulated for safety and cost reasons and software can help in that regard.

Implementing experiential learning in an object-oriented programming course is accomplished quite easily through the use of programming assignments. These assignments need to engage the learner in both object-oriented design and implementation of the design in order to be effective. Examples of similar implementations including simulations can be provided within the course content. Structuring these examples within a hypertext system that provides question and answering capabilities and web-like interconnectedness should be an effective instructional design approach through its ability to scaffold and articulate the content. Scaffolding and articulating are ways in which a learning environment can be designed to support cognitive apprenticeship, a constructivist model of instruction which makes thinking visible (Collins, Brown, & Holum, 1991). The integration of these concepts into the set of content-learner interactions should provide an improvement in learners' understanding and also offset the requirement for the instructor-learner interaction.

Incidental Learning

Learning often takes place when we least expect it to. The physics we learn by playing football or the geography we learn when we ride a bike are examples of incidental learning. Schank and Cleary (1995) suggest that schools could exploit the use of incidental learning in lieu of memorization of lists of facts. Facts should be incidentally learned within the context of something the student naturally wants to learn. Students should be allowed to adopt goals and be provided with materials that will cause them to pick up the desired information in passing. Course designers should develop content and transmission mechanisms which integrate factual knowledge that can be learned in a natural way.

It has been demonstrated that electronic mailing lists can provide an effective mechanism that supports incidental learning. In a study on the use of electronic mailing lists (Collins and Berge, 1996), when students were asked to identify how they learned, 29% of the respondents stated that their learning from the mailing list was unintentional and 53% of the respondents stated that it was a combination of unintentional and intentional learning. This model can easily be adapted to any type of course including one in object-oriented programming.

Learning by Reflection

Dewey (1910) defined reflection as "an active persistent and careful consideration of any belief or supposed form of knowledge in the light of the grounds that support it and the further conclusion to which it tends." Reflection is also one of the four stages in Kolb's (1984) learning model. Learners often learn when reflecting upon an assignment, generating a hypothesis, and presenting it to others for discussion. The process of reflecting enables one to learn from both their success and failures.

Schank and Cleary (1995) suggest that teachers can provide this mechanism by asking insightful questions for refection and then helping the learner identify the strengths and weaknesses in their arguments. The teacher can challenge the learner to think more deeply about the topic. They can test the reasoning for an answer and not just the answer. Guzdial (2001) uses reflective learning in a course where students use a collaborative website. The reflection occurs through peer review in a shared web space. Additionally a case library supporting an Objects and Design class in computer science is being developed by Guzdial which houses assignments with commentary for use by future students. This is an excellent example of how we can utilize technology to support learning by reflection.

Learning by Exploring

Learning-by-exploring means enabling the students to pursue their personal interests. This implies that students have access to multiple opinions and multiple experts, not just the teacher. Schank and Cleary (1995) suggest that the research-and-report model is not sufficient for this mechanism. They suggest that the expert knowledge should be easily obtainable and not embedded in a difficult-to-find paper at the library. They suggest the creation of easily-explorable video or text databases to support this mechanism.

One well accepted model for learning by exploring is a WebQuest. A WebQuest is a learning strategy developed by Professor Bernie Dodge in 1995 (March and Ozline.com, 2004). A WebQuest utilizes technologies associated with the World Wide Web and the Internet to support cooperative and active learning methods. A survey of currently implemented WebQuests in computer science and computer technology is provided in a review by Schmidt (2002).

Integration of the Teaching Architectures

Schank and Cleary (1995) suggest that these teaching architectures can be pulled together by using a goal-directed learning strategy. As in life, students have goals, they formulate plans that they believe will help them achieve these goals, and they decide what actions to take in order to advance those plans. When learning is motivated by personal goals, that is, is goal-directed, students are eager to respond because the learning is natural to them. Both intentional and incidental learning will occur naturally during goal-directed learning. For example, athletes often become very knowledgeable about nutrition in their attempt to improve their performance. As teachers we need to allow for students to pursue personal goals in order to generate student interest and motivation. This implies a redesign of the skill-centered courses to what Schank and Cleary refer to as Goal-Based Scenarios (GBS) (Schank and Cleary, 1995; Schank, 1996). A well designed GBS negotiates between the desires of students and those of the course designers. It provides students with the ability to pursue a clearly stated, interesting goal. A well designed GBS provides a transmission mechanism that effectively utilizes the learning architectures listed above, keeps the student motivated towards their goals, and enables the student to capture the necessary skills and outcomes required by the course.

In order to meet the requirements for a GBS, course interactions should avoid a drill-and-practice orientation. These interactions should directly support the learning architectures listed earlier such as learning-by-doing, natural learning, and learning by exploring. Finally, from an economical perspective the instructional design should minimize the instructor-student interaction. Providing students with the mechanisms to ask and answer their own questions based upon their own goals within the context of the course is a much preferable method.

CFT Grounded Hypertext Videos

This section presents the research on the use of hypertext video lecture recordings in order to enhance student learning. Hypertext videos are videos that are indexed through hyperlinks. This capability provides the learner with the ability to crisscross the video in an order which is controlled by the learner and not by the interface. Hypertext videos are an example of how we can reorganize instructional content to meet the goals of the learner (Schank and Cleary, 1995).

Videotape versus Digital Format

One of the more difficult components to capture in a lecture-oriented face-to-face course is the actual presentation of the lecture material by the teacher. The difficulty is not necessarily capturing the lecture in and of itself, but it is the capturing of the teacher-student interaction within the lecture. Assuming for the moment that the teacher-student interaction is a separate component as in the course interaction model, a simple videotaping of the lecture might be suitable (Russell, 1999). This taped lecture can be distributed to the student via mail and returned in some cyclic basis. However there are significant production and dissemination costs included in this model. Some of the obvious advantages of this model include that the lecture can be reviewed multiple times, and it is available at any time.

An alternative to the videotaped model is to record the lecture using computing technology. Within this model there are several alternatives. The viability of these alternatives is dependent upon the technical expertise of both the teacher and his or her support staff, on copyright issues, on technology, and bandwidth available to both the instructor and students. Recent qualitative and anecdotal studies show that effective lecture material can be created and delivered digitally by the teacher with limited development effort (Lindsey, 2003). Typically this material involves the use of streaming video along with visual slides that track with the lecture. However, due to the anecdotal nature of the studies, it is difficult to determine just how effective the lecture material is. Other studies have noted that the use of video clips within a content management system aligned with academic content standards, a form of computer-mediated instruction, has been shown to improve student achievement (Boster, Meyer, Roberto, & Inge, 2002; Reed, 2003). Mulford (2001) provides a detailed description of hardware and software requirements to do the recording yourself. Even if we are to assume that this model is simple enough to learn and deploy, it does not capture the teacher-learner interactions that may occur within a face-to-face lecture. It can be used, though, to supplement the course content, and in that respect, it offers significant benefits.

Digital videos offer the same advantages as those of videotapes. These advantages include the capability to be paused and replayed, they are available at time, and they are independent of the classroom. Additionally, because digital videos can be transmitted over the internet, there are no postage costs or time delays involved with getting the recorded video to the student.

Digital Hypertext Format

One of the more intriguing alternatives in lecture capturing is described briefly by Lynch (2002). In this alternative, an integrated software and hardware solution is used to record the lecture. These solutions have the ability to record a PowerPoint™ lecture integrated with the voice and video of the presentation. This provides an additional quality and advantage to the recorded media in that the learner has direct control over the lecture content through a software interface. This additional flexibility provides for repetition and review of the lecture with an improved interface over that of a videotape in that it allows for random access to the lecture content which a taped solution does not. It would seem that this additional flexibility would improve the learner-content interaction component of a course and have a positive impact on a student's achievement. Lynch reports that students were able to master material in self-paced study using this type of technology. Lynch notes that grades were higher when students were provided the digitally recorded video lectures as opposed to that of face-to-face lectures; however, Lynch provides no statistical evidence.

CFT Grounded Implementations of Hypertext Systems

This section describes three alternative implementations of hypertext systems which are ground in CFT. Case-based hypertext systems support knowledge construction by providing multiple perspectives with the ability to crisscross the landscape of the case-base with guided support (Nelson, 1994). These alternatives illustrate how the concepts of CFT can be implemented from an instructional design perspective in order to support student learning.

A CBR Hypertext Example without an Automated Reasoner

An interesting example of how a hypertext environment can be implemented as a case-base reasoning system without an automated reasoner is clearly illustrated by the hyperbook *Engines for Educations* (Schank and Cleary, 1995). The hyperbook describes a learning theory that suggests learners build knowledge through reasoning from what they remember about their past experiences. What makes this hyperbook unique is the ability to crisscross the material through multiple paths, a strategy that is a requirement of Cognitive Flexibility Theory. By providing alternate indexes such as categorized lists, content outlines, and content related to reader objectives into the same body of material, Schank and Cleary have cleverly incorporated the techniques described by Spiro et al. (1992). Additionally, examples or cases have been provided as a support mechanism throughout the text. They are provided as remindings for the learner to use in their construction of new knowledge. This has all been accomplished with no automated reasoner, which makes it all the more intriguing because of its simplicity to implement.

A CBR Hypermedia Example with an Automated Reasoner

Creanimate is a case-based teaching system that uses hypermedia and an automated reasoner to teach children about biology (Edelson, 1998). It was designed to help elementary school-aged children learn about how animals adapt with particular emphasis on physical features and how the physical features enable them to survive. The case-based reasoning system asks the students questions about creating new animals based upon the remindings. These remindings might suggest why some animals have wings and how wings help them adapt and survive in their environment. It is left up to the learner to create a new animal with the appropriate features that will allow them to survive. The learner can crisscross through the case-base creating new animals and learn biology in the process. Although this is not a web-based implementation in hypertext, it provides us with an alternative that illustrates how a case-based reasoning system can help students learn.

A Web-based CBR Hypertext Example with an Automated Reasoner

A final CBR system example which uses a web-based hypertext interface but also includes some built-in reasoning is an application that is currently in the "proof of concept" stage of development called *CBRubric* (Schmidt and Lalonde, 2004). This application deals with the development and use of assessment rubrics. These rubrics are instruments used to measure outcomes of student performance in relationship to goals and objectives of a particular educational unit or course. Typically these rubrics can be broken down and indexed by a skill, dimension of the skill, a ranking of the skill, and a related textual description of the rank. Figure 4 depicts an example.

	UNSATISFACTORY	**BASIC**	**PROFICIENT**
Writing			
Responsive To Article	Presents a response to the ideas presented in the article that is surface and/or lacks in-depth engagement of the ideas presented. Weak presentation of the relationship between the ideas presented in the article and an education related issue.	Presents a logical response to the ideas presented in the article, and an exploration of the impact of these ideas on an education related issue that is fairly well supported.	Presents an insightful, logical, and compelling response to the ideas presented in the article and a well-supported exploration of the impact of these ideas on an education related issue.

Figure 4. Responsive to Article Dimension for Writing Rubric

In this example there is one skill listed, *Writing*, with one dimension called *Responsive to Article*. It has three rankings including unsatisfactory, basic, and proficient. The relationship between the ranks, skills, and dimensions are described in the text. These are referred to as benchmarks.

Assessment of students and the use of the assessment knowledge are very complex and are ill-structured. To get an idea of this, just ask a couple of different people in education about their thoughts on it, and you will get many different answers.

Thus, the building of a rubric and its use for assessment is a great opportunity for us to use a case-based reasoning system to support this process.

To get an idea of how this system is currently being implemented so that it conforms to the CFT principles set forth by Spiro and Jacobson (1995) we need to consider the indexing mechanisms, accessing strategies, alternative points of view, and the crisscrossing mechanisms for this application. Indexing structures have been developed which includes indexes on the skill, dimension, rank, and benchmark. Additionally since rubrics are tied to units or courses, these have been also been included as indexes. Sample queries into the case-base provide lists such as give me a list of the rubrics that test writing skills and are math based. Accessing and implementation are through a hyperlinked document with an underlying relational database. The CBR system also provides the ability to adapt and incorporate new knowledge. Thus, as new rubrics are generated, the system allows for the storage and retrieval of the new knowledge.

Future enhancements to CBRubric will provide indexes that support access based upon the user interest, that is, student, teacher, or administrator interest. For example a student might have a different interest in the assessment of writing in math versus that of a teacher. The student may want to look for tips on how to meet a goal or objective whereas a teacher may be interested in the how to evaluate an artifact based on the rubric. Thus the system should provide for that. Additionally, improvements in searching the case-base will be upgraded so that any combination of the indexes may be searched.

Implications and Concluding Remarks

CBR is a principle of constructivism that can be used to support advanced learning in an ill-structured domain. Several alternative methods such as EBR, MBR, ABR, and CBR are based on the CBR paradigm. CBR provides an environment that allows for the crisscrossing of the knowledge content, and it underlies CFT. Hypertext learning environments can be established that incorporate CBR in a variety of ways. Examples can be provided in an automated way through an advanced search, reuse, and revise mechanism, or they can be provided with clever hypertext indexing schemes. The teaching architectures identified by Schank and Cleary (1995) provide us with the theoretical foundation and support for using hypertext videos to supplement for course content. Finally, CBRubric (Schmidt and Lalonde, 2004) is a CBR application which exploits CFT by providing scaffolding for the development of assessment rubrics. It provides us with a simple foundation to build alternative CBR systems that can support learning in other educational domains and suggests further research in this area.

Implications of this literature review suggested that the use of indexed hypertext videos as part of an online course that has a theory-based design should work as well or better than any other medium in the transmission of the course content. The literature review also suggested that using hypertext videos in this way may not be good enough. The pilot study referred to in chapter one illustrated some of the problems and issues. However, by supplementing an online course with a CBR application grounded in CFT that supports additional scaffolding for the complex and ill-structured topics, we may be able to overcome some of the limitations of the hypertext video transmission model. These were the driving issues for doing this study.

Chapter III: Methods

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases would improve a student's understanding of the more difficult concepts in an object-oriented programming course. The second part of this study examined the relationships between the dependent variable student performance with independent variables of motivation, background knowledge, and student attitudes towards the case-based hypertext learning tool. This chapter describes the methods and procedures that were used in this study as well as details about the case-based hypertext learning tool.

Research Design

This study employed non-parametric and correlational analysis research designs. The first part of the study involved characterizing the sample based on learner competency assessment questions categorized according to Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956). Factor *A* was defined as instructional support (case-based hypertext learning tool versus lecture notes only) and factor *B* was defined as the question type (simple assessment questions that measure the lower levels of learner competency and complex assessment questions that measure the higher levels of learner competency). The experimental design is shown in Figure 5. In the experimental design depicted in Figure 5 Group 1 and Group 2 refer to the groups of students randomly selected from two sections of Object-Oriented Programming 1. Object Design and Inheritance are the two content areas that had instructional support. The treatments X(CBA) and X(LNO) refer to the case-based hypertext learning tool support and the lecture notes only support (case-based reasoning assisted, i.e. CBA, is also used in place of the term case-based hypertext learning environment. Essentially within the context of this study, both terms have the same meaning). Posttest 1 and Posttest 2 refer to the two posttests that will be given.

	Object Design	Posttest 1	Inheritance	Posttest 2
Group 1	X(CBA)	O	X(LNO)	O
Group 2	X(LNO)	O	X(CBA)	O

Figure 5 Experimental Design depicting Groups, Treatments, and Observations

The second part of the study involved performing a correlational analysis of the dependent variable student performance with the independent variables of motivation, background knowledge, and student attitudes towards the case-based hypertext learning tool. Measurements on motivation and student attitude were self-reported ranked and narrative data. Background knowledge was ranked data.

A quantitative research paradigm was chosen primarily based on the nature of the research questions. In particular this research looked for relationships between multiple variables as well as identifying treatment effects, and this is best handled through quantitative analysis. Using correlational analysis as well as non-parametric designs helped limit the risk of a small sample size. The instruments used to collect the data for analysis were known up front and, for the most part, had been validated in prior research.

What follows is a restatement of the research questions addressed by this study, including the researcher's hypotheses. Question three has no hypothesis because there was no expectation on the findings; it is simply an issue of curiosity. A detailed description of the sample, treatments, and measures is provided in following sections.

1. Is there a statistically significant difference in the performance on simple questions between the Case-Based Reasoning Assisted (CBA) group and the Lecture Notes Only (LNO) group?

   Null Hypothesis: There will be no significant difference in the student performance on the simple questions between the CBA group and the LNO group.

   Alternative Hypothesis: There will be a significant difference in the student performance on the simple questions between the CBA group and the LNO group.

2. Is there a statistically significant difference in the performance on complex questions between the CBA group and the LNO group?

   Null Hypothesis: There will be no significant difference in the student performance on the complex questions between the CBA group and the LNO group.

   Alternative Hypothesis: There will be a significant difference in the student performance on the complex questions between the CBA group and the LNO group.

3. How is student performance on the measures of learner competencies related to motivation, background knowledge, and attitudes towards the case-based hypertext learning tool?

Sample

Two sections of the Object-Oriented Programming 1 (OOP-1) course in Spring 2005 were used to represent the sample. Only two sections were used because there were only two sections of OOP-1 offered at the study site. OOP-1 was an introductory course in Java which emphasized object-oriented programming and design. It was the second required programming course in both the Bachelor of Science degree in Computer Information Systems and the Bachelor of Arts degree in Computer Information Science at an NCAA, Division II university located in the Midwest region of the United States. Students typically take this course during the second semester of their freshman year. Of the two sections, one was face-to-face and the other was an online distance-education course. During the period of the quasi-experiment all students were attending the course as if they were in the online section. The experiment began the seventh week of class and ran for three weeks.

Permission to use human subjects was obtained from the Kansas State University's Institutional Review Board as well as the Institutional Review Board of the study site prior to the beginning of the study. Students were asked to complete an informed consent form (Appendix A) prior to the start of the study. All subject data will remain confidential. Subject names were used to match the post test scores across the experiment, background coursework grades, and survey data. Following this matching process, arbitrary numbers were used to identify and refer to subjects.

Twenty-one students were initially enrolled in the two sections of Object-Oriented Programming 1. Of these, 11 students were enrolled in the online section and 10 students were enrolled in the face-to-face section. Prior to the beginning of the study, five students dropped the course. Of the remaining 16 students, all signed the consent form agreeing to participate in the study. These 16 students were randomly assigned to one of two treatment groups: Group 1 or Group 2. This random assignment process was handled as follows: A listing of student names from both courses were loaded into a Microsoft Excel spreadsheet. The students were then sorted by last name. Finally, the randomization function built within Microsoft Excel was applied providing each student with a randomly assigned unique integer number. The students were then sorted by that uniquely assigned number in ascending order. The first eight of these students were assigned to Group 1 and the second eight were assigned to Group 2. Before the completion of the experiment one student from Group 1 was dropped from the study because of the student failed to take both posttests.

The use of two types of instructional support, i.e. factor *A,* was varied over two different complex topic areas during the experiment. These topic areas included object design and inheritance. Object design and inheritance were chosen because of their similarity in their levels of complexity.

Learner competency was assessed based on the questions from each of the two posttests. The questions were divided into two groups, that is, factor *B*, simple questions and complex questions. Complex questions were those questions which require higher-order thinking whereas the simple questions were more related to facts, recall, and straight forward application. All students in both groups were provided with the identical assessment questions.

Treatments

Lecture content to members of both treatment groups 1 and 2 was provided in the form of hypertext videos that were administered through WebCT. The recordings were developed using sofTV. Each of these recordings was placed into WebCT and integrated through a hypertext document. Both groups also shared an online space in WebCT. All lecture-notes, online discussions, and homework assignments were also provided and administered through WebCT to both groups. Email was handled externally using the study sites' email system.

Two complex and ill-structured content areas had additional instructional support through a case-based reasoning tool called *CBJava* (Schmidt, 2004). These content areas were object design and inheritance. During the coverage of object design, Group 1 was required to use the CBJava tool, that is, the CBA treatment. Group 2 received no assistance from CBJava during this period, that is, the LNO (lecture notes only) treatment. After completing the coverage of object design, Posttest 1 was given to both groups. Posttest 1 (Appendix B) contained both simple and complex assessment questions related to object design. The duration for this part of the experiment was one and one half weeks culminating with the Posttest 1.

Inheritance was covered immediately following the unit on object design. During the coverage of inheritance, Group 2 was required to use the CBJava tool, that is, the CBA treatment. Group 1 received no assistance from CBJava during that period, that is, the LNO treatment. After completing the coverage of inheritance, Posttest 2 was given to both groups. Posttest 2 (Appendix C) contained both simple and complex assessment questions related to inheritance. The duration for this part of the experiment was one and one half weeks culminating with the Posttest 2.

In support of the correlational analysis portion of the study additional supporting data were collected. Learning motivation data were collected using the Motivation Survey (Appendix D). This survey was administered immediately following Posttest 1 and Posttest 2 to those students who received the LNO treatment during that period. Student attitudes towards the CBJava tool were also collected through the Student Attitudes towards CBJava Survey (Appendix E). This survey was administered immediately following Posttest 1 and Posttest 2 to those students who received the CBA treatment during that period. The survey data were used to assist in the interpretation of the quantitative findings. Background knowledge measurements were obtained for all students from their university transcripts.

Both posttests and surveys were administered on the study sites' campus. Additionally these tests and surveys were proctored by a faculty member at the study site who was not the researcher (myself). This faculty member coded both the tests and the surveys in order to protect anonymity during the study. Both posttests were scored by the researcher before they were matched back to the student in order to minimize bias. Survey results were withheld from the researcher until after both posttests were scored. This was done to minimize influences on the treatments.

Measures

Motivation: pertains to why a student wants or desires to learn. Motivation can be either intrinsic: to undertake an activity "for its own sake, for the enjoyment it provides, the learning it permits, or the feelings of accomplishment it evokes" or extrinsic: to undertake an activity "*in order to* obtain some reward or avoid some punishment external to the activity itself," such as grades, stickers, or teacher approval Lepper (1988, p. 292). A motivation survey (Appendix D) was developed to measure motivation in an attempt to classify the student as either intrinsically motivated or extrinsically motivated.

For the purposes of this study intrinsic motivation is defined as the mean score on items 1-4 of the motivation survey and, extrinsic motivation is defined as the mean score on items 6-10 of the motivation survey. Question 5 was a social motivation question (Jenkins, 2001) and was inadvertently included in the survey; it was not used in the analysis. Question 11 was an open-ended question used to capture any additional motivation related data that the student wished to provide. Similar questionnaires have been developed (Jenkins, 2001; Mitchell, Sheard, and Markham, 2000; Wilson and Braun, 1985) to measure motivation in computer science courses. In order to insure validity and reliability of this measurement these were used to inform the development of the motivation survey. Additionally, the survey was reviewed and critiqued by an expert in survey development.

Learner Competency: Two posttests, Posttest 1 (Appendix B) and Posttest 2 (Appendix C), were given immediately following the treatment conditions. Each test contained a set of questions covering the content areas addressed during the respective period in the study. These questions were categorized according to Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956).

The categorization of the test questions went as follows. Two categories of questions were created, simple questions and complex questions. Simple questions were those questions which measured the learning objectives of knowledge, comprehension, and application. The complex questions were those questions which measured the learning objectives of analysis, synthesis, and evaluation. A set of candidate questions for each of the two tests was generated by the researcher who pulled candidate questions from the normal assessment tests given in previous semesters. These candidate questions were then provided to two other faculty members who had previous experience in teaching a Java programming course. Each of these faculty members as well as the researcher classified the questions as either simple or complex according to Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956). Questions were then classified by majority vote as either simple or complex. Additionally, questions which were unclear were either clarified or dropped as candidate questions. Below is a description of the learning objectives according to Bloom's taxonomy.

1. Knowledge: pertains to recall of specifics, memorization, recognizing, enumerating, identifying, or defining.

2. Comprehension: pertains to the grasping of the meaning of informational materials such as providing examples, explaining, classifying, or generalizing.

3. Application: includes the ability to articulate, construct, make operational, discover, and predict solutions to problems that have a best answer using previously learned information.

4. Analysis: pertains to the ability to break down a problem into its constituent parts so as to identify its motives or causes. It is the ability to analyze, dissect, and discover relationships.

5. Synthesis: deals with the ability to construct a solution by taking individual parts and combining them. It is the ability to build, construct, and invent new solutions.

6. Evaluation: pertains to the ability to criticize, defend, and make judgments on work based upon predefined criteria.

Both posttests had an identical format and ordering of questions. There were 12 true/false questions, seven multiple choice questions, and six short answer questions. Of the 12 true/false questions, the first 11 questions were categorized as simple questions and the twelfth one was categorized as complex. All of the multiple choice questions were categorized as simple questions. Finally, all of the short answer questions were categorized as complex.

Each of the posttests was worth a total of 50 points. Of these 50 points, the true/false questions were worth one point each, the multiple choice questions were worth two points each, and the short answer were worth four points each. Thus, the simple questions were worth 25 points and the complex questions were worth 25 points. A sub score for each question type was generated based on the total points scored on the related assessment questions.

Finally, although the short answer questions had a greater point value per question, partial credit was given for partially correct answers. In order to ensure consistency in grading, a list of reasons for the partial credit along with the amount of partial credit awarded was maintained and used as a guide for awarding points.

Background: deals with the specified course prerequisites that have been identified for entry into the course. The course perquisites included the successful completion of CM111 - Introduction to Structured Programming and PH110 - Logic for Computer Programming. Successful completion was defined as having scored a minimum of a C in the course. The data collected for this measurement were obtained from the student's transcript. These data included the letter grade for each of the prerequisite courses. Letter grades were recorded as ranked data using the following scale A = 4, B = 3, C = 2, D = 1, and F = 0.

Student Attitudes towards CBJava: pertains to what the students liked and disliked about the CBJava tool. A short Student Attitudes towards CBJava survey (Appendix E) was developed to collect student attitudes in an attempt to determine if there is a relationship between what the students thought about the tool and their performance on the related content areas. The survey included both closed ended questions that contained ranked responses based upon a five-point Likert scale as well as open ended questions. The survey questions were developed by the researcher and validated by another faculty member who was an expert in survey development. The survey was structured so that any bias towards negative or positive responses was minimized.

CBJava

CBJava is a case-based hypertext system that was developed to be used as an instructional content aid for students who are learning Java (Schmidt, 2004). The design of this system was similar to the hyperbook design used in the *Engines for Education* web site authored by Schank and Cleary (1995). In addition to being a hyperbook, this site provided students the ability to add their own examples. As these examples were added, an expert (in this case the researcher) rated the examples based upon quality and context. In this way a case-base of validated examples were made available to other students for further learning and research.

System Architecture

The foundational architecture of CBJava was the question and answer interface that sits on top of a relational data model providing both knowledge acquisition and user feedback. Details of the implementation are beyond the scope of this dissertation. However, the foundational architecture of CBJava did play an important role in the treatments. Therefore, a high level overview of the system's data architecture, user interface architecture, and knowledge acquisition architecture is provided below. Although the detailed research in the reliability and validity of using CBJava as a learning aid has not been researched, its design was based upon well accepted learning models including learning by example, learning by exploring, and Cognitive Flexibility Theory. Some validation of these components was handled through expert review. Modifications based upon feedback from the expert were incorporated into the design.

Data Architecture

CBJava's data architecture consisted of seven related entities which are depicted in Figure 6. In this model the data entities are represented by the boxes and their relationships are illustrated by the connecting lines. The direction of the relationship between two entities is illustrated by the dot. For example a TopicArea entity is related to many TopicAreaExample entities. The primary implementation and storage of the data content was within JSP (Java Server Pages) documents with the exception of the Example entity. The Example entity and its links was stored in a relational database and rendered for viewing by the CBJava application in HTML. Definitions of the entities are provided as follows:

1. <u>TopicArea</u>: refers to an instructional topic area within a content area of Java. For example within the content area of object design there are a set of supporting topic areas such as cohesion, coupling, accessor and mutator methods, etc.
2. <u>RealWorldProblem</u>: refers to a real world software design problem that can be solved using Java. An Automated Teller Machine is an example of a real world problem.

3. ProblemExample: provides the relational mechanism between an example and a real world problem. An Example object can be used in support of solving many different real world problems.
4. ContentArea: refers to an instructional subject area within Java. In particular there were three content areas that were represented with the initial version of CBJava. These content areas included decisions, object design and inheritance. The decisions content area was very limited with its sole purpose being that of a student training area.
5. ContentAreaTopic: provides the relationship between a content area and a topic area. A topic area may support more that one content area. For example the topic area of cohesion applies both to the content areas of inheritance and object design.
6. TopicAreaExample: provides the relationship between a topic area and its related examples. Examples may illustrate multiple topic area questions.

7. Example: refers to the description and implementation of a Java class which is used to offer an example of a particular answer to a question.

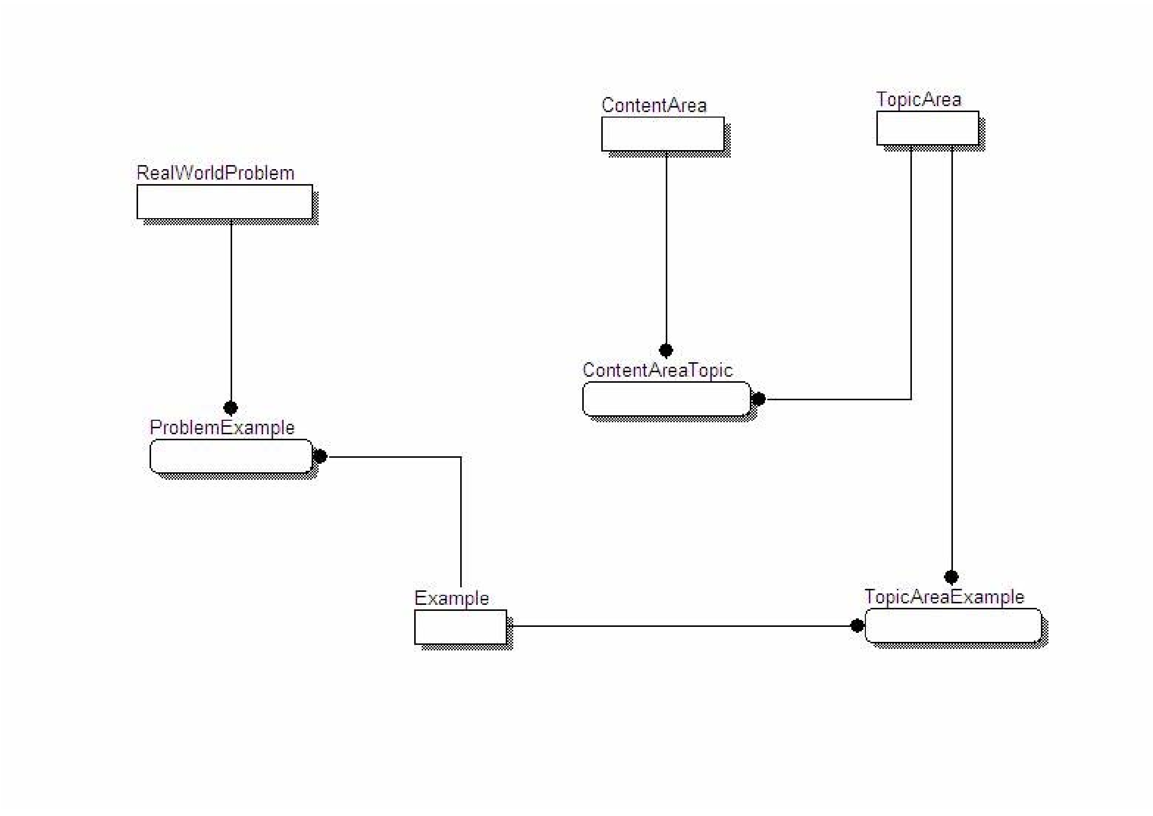
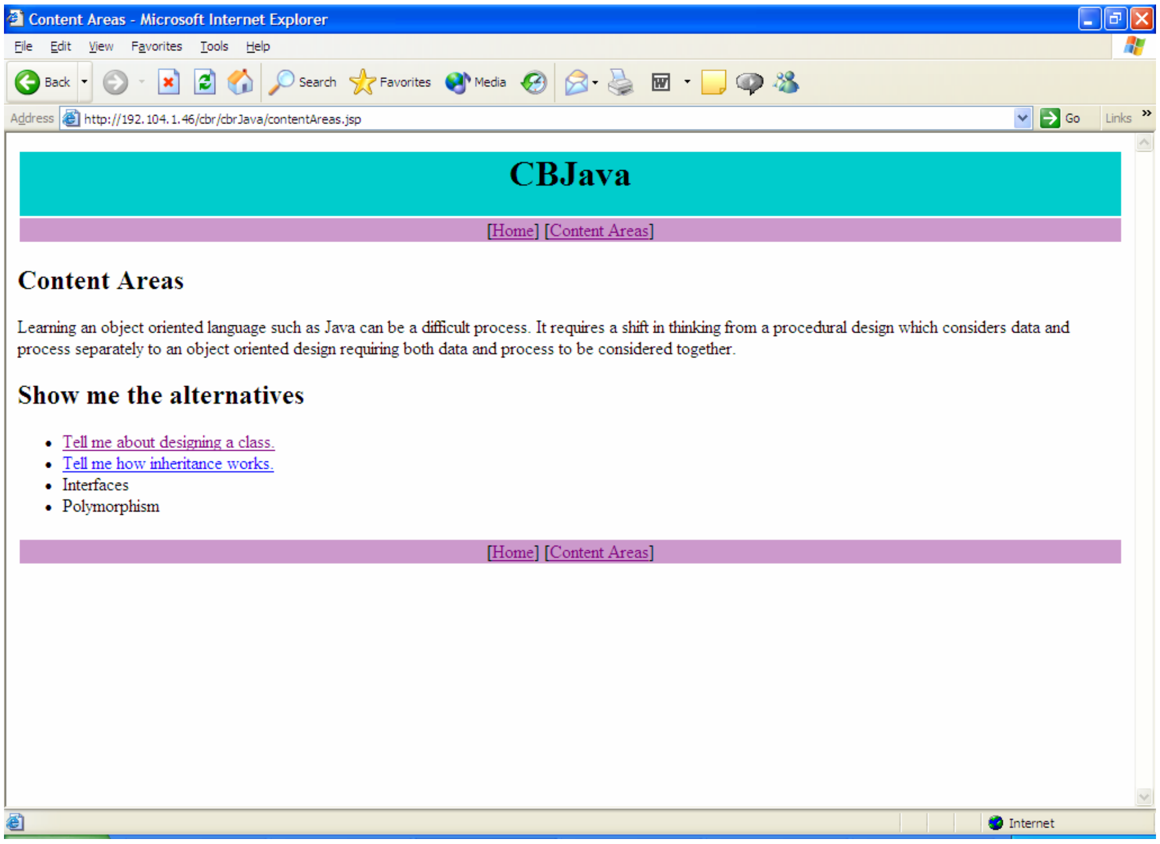


Figure 6. CBJava's Data Architecture

User Interface Architecture

CBJava's user interface was a hypertext interface implemented in JSP that is accessible on the World Wide Web. It supported a question and answer model through the clicking of links. For example: to pose a question about the content area of Object Design the student could click on a link with the question "Tell me about designing a class" (see Figure 7 below). The system responded with an answer by displaying the web page on object design (see Figure 8 below). The response page provided a brief discussion about Object Design with additional questions that could be posed. Examples could be accessed and additionally an interface was provided for the student to add their own examples.


Content Areas - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://192.104.1.46/cbr/cbrJava/contentAreas.jsp
Go
Links
Internet

CBJava

[Home] [Content Areas]

Content Areas

Learning an object oriented language such as Java can be a difficult process. It requires a shift in thinking from a procedural design which considers data and process separately to an object oriented design requiring both data and process to be considered together.

Show me the alternatives

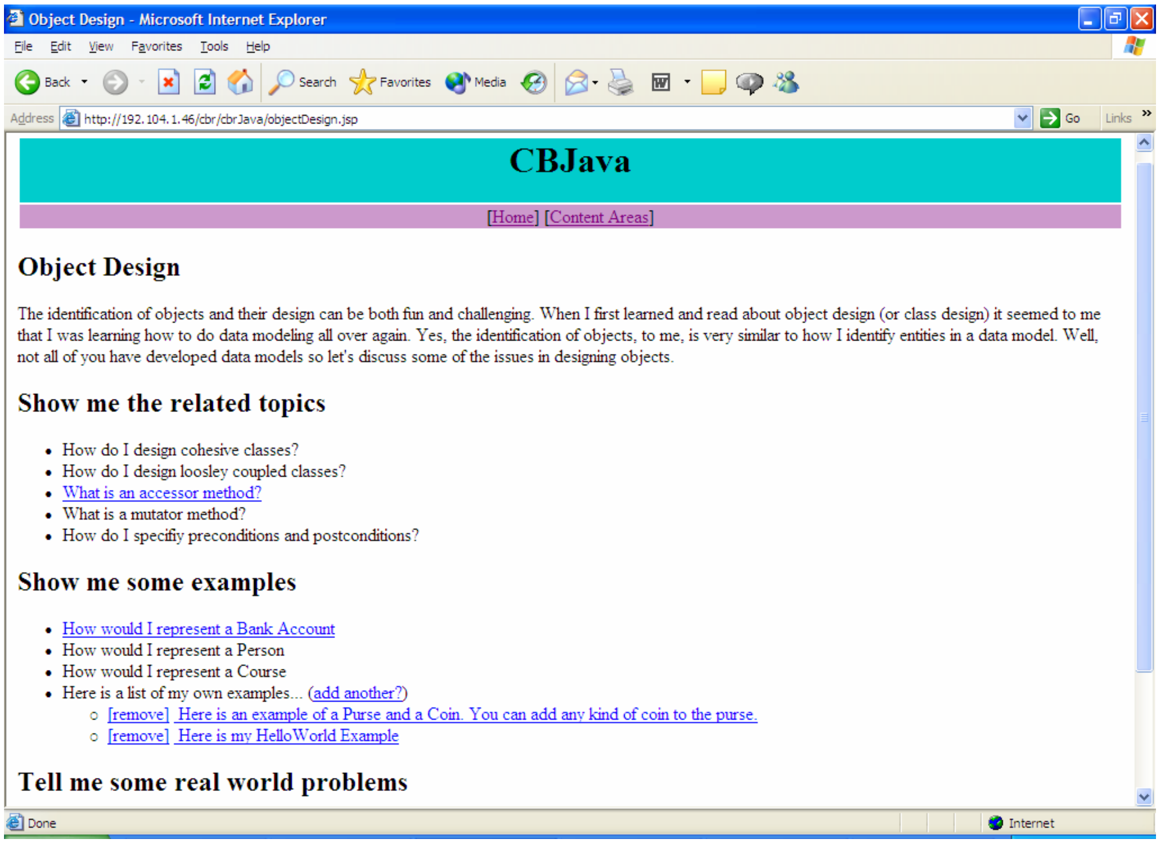
- Tell me about designing a class.
- Tell me how inheritance works.
- Interfaces
- Polymorphism

[Home] [Content Areas]

Figure 7. CBJava Content Areas Web Page

Knowledge Acquisition Architecture

The ability for the learner to add content to the site provided an additional feature which is not available in the hypertext design used in the Engines for Education web site (Schank and Cleary, 1995). Reasoning capabilities of CBJava were managed through the initial indexing of the examples by the learner and the revision and classifications by the expert. Examples were classified as either well defined, adequate, or needs work by the expert (the researcher) as part of the repair and revision process.


Object Design - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.104.1.46/cbr/cbrJava/objectDesign.jsp
Go Links
Done
Internet

CBJava

[Home] [Content Areas]

Object Design

The identification of objects and their design can be both fun and challenging. When I first learned and read about object design (or class design) it seemed to me that I was learning how to do data modeling all over again. Yes, the identification of objects, to me, is very similar to how I identify entities in a data model. Well, not all of you have developed data models so let's discuss some of the issues in designing objects.

Show me the related topics

- How do I design cohesive classes?
- How do I design loosley coupled classes?
- What is an accessor method?
- What is a mutator method?
- How do I specifiy preconditions and postconditions?

Show me some examples

- How would I represent a Bank Account
- How would I represent a Person
- How would I represent a Course
- Here is a list of my own examples... (add another?)
  - [remove] Here is an example of a Purse and a Coin. You can add any kind of coin to the purse.
  - [remove] Here is my HelloWorld Example

Tell me some real world problems

Figure 8. CBJava Object Design Web Page

CBJava's primary knowledge acquisition process involved the submission of new Java examples by the student and expert review and validation performed by the instructor (the researcher). Indexing of the example was performed by the student through a Web interface. During the study only one of the content areas within CBJava was open at a time. Thus the indexing of the example was limited to that particular content area. For example those students who were given the CBA treatment during the first period of the study could only index their examples under object design. Those students had no access to the inheritance content area. During the second period of the study those students who were given the CBA treatment could only index their examples under inheritance. Again, those students had no access to the object design content area.

At the time the example was submitted the example had a note stating that it has not been validated. On a daily basis the instructor reviewed the submissions and either accepted the submission or revised it. In the revision cycle, the instructor identified the improvements and classified the original example. Both the before (as submitted) and the after (post review) will be retained.

The knowledge acquisition process implemented in CBJava was fairly simple although it did demand feedback from a human expert. This is not unlike what happens in a real life context. It is problematic for a computer program to determine the efficacy of a Java class. Thus, this simple implementation of knowledge acquisition might be the most effective as well. Most importantly it can be used as to test the viability of case-based hypertext learning tool (factor *B* in this research design).

Data Analysis

Effects of CBJava: Two types of non-parametric tests were performed because of the small sample size, and a parametric test was performed in order to identify carryover effects. A Wilcoxon Signed Ranks test was performed at each level of instructional support at $p < .05$ level of significance. The Wilcoxon Signed Ranks test is a non-parametric test that was used because it can measure two groups of matched individuals (Huck, 2004). This test was performed for each question type: complex, simple, and both. Matched pairs were created by matching the performance measurements of the same student across the two content areas: object design and inheritance.

In order to determine if there were carryover effects between the two periods of the study, a parametric test was conducted immediately following the Wilcoxon Signed Ranks test. A 2×2 mixed-design analysis of variance (ANOVA) at $p < .05$ level of significance was calculated to examine the carryover effects of the CBA treatment on Group 1's test scores.

Following the test for carryover effects, A Mann-Whitney *U* test was performed at each level of question type at $p < .05$ level of significance. The Mann-Whitney *U* test is a non-parametric test that can be used to compare two independent samples (Huck, 2004). This test was performed for each content area (object design and inheritance) pending the results of the test for carryover effects. It compared the performances of students in Group 1 to students in Group 2.

Correlational Analysis: A correlational analysis was performed in order to determine if and how student performance on test questions is dependent upon the variables of motivation, background knowledge, and student attitudes towards the CBJava tool. Three separate sets of bivariate correlational tests were performed. Each of these variables was composed of one or more measures that were either collected through a survey or through the student's university transcript. Using these measurements, a Spearman *rho* correlation coefficient was calculated between the student's test scores in each of the content areas (object design and inheritance) broken out by question type (simple, complex, and total) and the particular measurement. This process was repeated for each of the three variables.

Additionally the open ended questions contained in the Motivation Survey and Student Attitudes towards CBJava Survey were documented and synthesized. The open ended questions were used primarily for gathering additional motivation data, feedback to support additional research into the development of the CBJava tool, and additional perspectives on the usefulness of CBJava.

Human Subject Considerations

All students received credit for using the CBJava tool as an incentive however this score was not factored into the study. To receive credit the student was required to create a content area example and post it to CBJava. All postings were anonymous to the other students but could be traced backed to the student by the researcher. Before receiving the CBA (CBJava assisted) treatment, students were provided with a training area within CBJava that was not part of the study. The training area set up for the students was composed of the decisions content area. Students were required to sign on to the CBJava site and post an example to the training area. This training occurred two weeks prior to the actual study. All students took part in this training.

Although the data collected in this study was primarily in the context of normal coursework, permission was requested (Appendix A) of the student for their participation in the study. All raw data including surveys and test scores will be kept in full confidence by the researcher.

Summary

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases (CBJava) would improve a student's understanding of the more difficult concepts in an object-oriented programming course. The second part of this study examined the relationships between the dependent variable student performance with independent variables of motivation, background knowledge, and student attitudes towards CBJava. Subjects for the study were randomly assigned from two sections of the Object-Oriented Programming 1 course. Posttests were used to measure the affects of CBJava on learner competency. Background knowledge was collected through student transcripts. Motivation and student attitudes towards CBJava data were collected from surveys. Open ended survey questions were documented and synthesized. The Wilcoxon Signed Ranks test, Mann-Whitney *U* test, mixed-design

ANOVA, and Spearman *rho* correlations were used to analyze the quantitative data. All data were collected during the Spring 2005 semester.

Chapter IV: Results

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases will improve a student's understanding of the more difficult concepts in an object-oriented programming course. The second part of the study involved performing a correlational analysis of the dependent variable student performance with the independent variables of motivation, background knowledge, and student attitudes toward the case-based hypertext learning tool. As previously presented, the research questions are:

1. Is there a statistically significant difference in the performance on simple questions between the Case-Based Reasoning Assisted (CBA) group and the Lecture Notes Only (LNO) group?
2. Is there a statistically significant difference in the performance on complex questions between the CBA group and the LNO group?
3. How is student performance on the measures of learner competencies related to motivation, background knowledge, and attitudes towards the case-based hypertext learning tool?

All data for this research were collected during the Spring 2005 semester at the study site. Subjects were randomly assigned to one of two groups (Group 1 and Group 2) from two sections of Object-Oriented Programming 1, that is, students assigned to each Group for experimental purposes were from both classes. Subjects completed the object design test (Posttest 1) at the end of time period one and the inheritance test (Posttest 2) at the end of time period two. Period one ran for one and a half weeks beginning at the seventh week of class. During period 1 students assigned to Group 1 received the CBA treatment, and students assigned to Group 2 received the LNO treatment. The CBA treatment refers to providing the students' access to CBJava for content area covered during the period, and the LNO treatment refers to providing the students only with the lecture notes for the content area covered during the period. Period two followed immediately after period one and, it also ran for one and a half weeks.

During period two students assigned to Group 1 received the LNO treatment and, students assigned to Group 2 received the CBA treatment. Additionally, at the end of period one, those subjects assigned to Group 1 completed the Student Attitudes towards CBJava Survey (Appendix E) and those students assigned to Group 2 completed the Motivation Survey (Appendix D). At the end of period two, the surveys were reversed, that is, the students assigned to Group 1 completed the Motivation Survey and the Students assigned to Group 2 completed the Student Attitudes towards CBJava Survey. It made sense for the students to fill out the Student Attitudes towards CBJava Survey immediately following their use of the tool while the use of the tool was fresh in their minds. However, the timing of the Motivation Survey was strictly based upon student availability and the desire to not overload them with more than one survey at a time.

This chapter presents an analysis of the data. A Wilcoxon Signed Ranks test, Mann-Whitney *U* test, and a mixed-design analysis of variance (ANOVA) were used to test for significant differences between those subjects who received the CBA treatment and those who received the LNO treatment. Spearman *rho* correlations, descriptive statistics, and a thematic analysis were used to investigate the relationships between background knowledge, motivation, and student attitudes towards the CBJava tool with respect to each of the posttests (Posttest 1 and Posttest 2) categorized by question type (Simple and Complex).

Sample Analysis

Of the 21 subjects who were initially enrolled in the only two sections of Object-Oriented Programming 1 offered at the study site, five subjects dropped the course. Sixteen subjects began the study. Of the 16, one subject was dropped after the first period due to failure to complete the posttest. Subjects were randomly assigned to one of the two groups (Group 1 and Group 2).

Effects of CBJava

In order to answer the first two research questions, two types of non-parametric tests were performed on the sample as prescribed in the experimental design. A third test, a parametric test, was performed on the sample in order to better understand the results of the first two tests.

Descriptive statistics including means and standard deviations for each content area (Object Design and Inheritance) separated by question type (Simple and Complex) for each group (Group 1 and Group 2) are provided in Table 3. Notice that the mean scores on the posttests (Posttest 1 covered Object Design and Posttest 2 covered Inheritance) for Group 1 are consistently higher than those in Group 2. A higher mean score represents a better performance on the posttest. The mean scores are the average number of total points for the group out of a maximum of 25 points for each part of the posttest (i.e., each posttest contained complex questions and simple questions each worth 25 points). Neither group did as well on the posttest covering inheritance (Posttest 2) as they did on Posttest 1.

Table 3. Descriptive Statistics for Test Scores for Question Type by Content Area by Group

Question Type × Content Area	Group 1			Group 2			Total		
	Mean	*N*	*SD*	Mean	*N*	*SD*	Mean	*N*	*SD*
Simple									
Obj. Des.	22.00	7	2.449	19.63	8	4.340	20.73	15	3.674
Inheritance	20.71	7	2.289	18.00	8	2.976	19.27	15	2.939
Complex									
Obj. Des.	22.71	7	1.890	17.88	8	3.563	20.13	15	3.758
Inheritance	20.57	7	3.505	15.63	8	3.852	17.93	15	4.383
Total									
Obj. Des.	44.71	7	3.450	37.50	8	5.682	40.87	15	5.927
Inheritance	41.29	7	5.282	33.63	8	5.476	37.20	15	6.527

Notes. Group 1 received the CBA treatment in period 1 (content: Object Design) and the LNO treatment in period 2. Group 2 received the CBA treatment in period 2 (content: Inheritance) and the LNO treatment in period 1. CBA treatment refers to providing the students' access to CBJava for the content area covered during the period, and the LNO treatment refers to providing the students only with the lecture notes for the content area covered during the period. The mean scores reported are average posttest scores broken out by question type (posttest 1 covered object design and posttest 2 covered inheritance).

Descriptive statistics including means and standard deviations for each level of instructional support treatment (CBA and LNO) separated by question type (Simple and Complex) for each group (Group 1 and Group 2) are provided in Table 4. Notice that the mean scores for Group 1 were consistently higher than Group 2 regardless of instructional support treatment.

Table 4. Descriptive Statistics for Test Scores for Question Type by Treatment by Group

Question Type × Treatment	Group 1			Group 2			Total		
	Mean	*N*	*SD*	Mean	*N*	SD	Mean	*N*	*SD*
Simple									
CBA	22.00	7	2.449	18.00	8	2.976	19.87	15	3.357
LNO	20.71	7	2.289	19.63	8	4.340	20.13	15	3.461
Complex									
CBA	22.71	7	1.890	15.63	8	3.852	18.93	15	4.728
LNO	20.57	7	3.505	17.88	8	3.563	19.13	15	3.681
Total									
CBA	44.714	7	3.450	33.625	8	5.476	38.800	15	7.272
LNO	41.286	7	5.283	37.500	8	5.682	39.267	15	5.650

The first non-parametric test performed was a Wilcoxon Signed Ranks test at each level of instructional support (CBA and LNO) at $p < .05$ level of significance. The Wilcoxon Signed Ranks test was used because of the small sample size to assess the significance of change within groups. It also allows for measuring two related samples of data generated by measuring the same people twice (Huck, 2004, p. 501). This test was performed for each question type (Simple and Complex). Matched pairs were created by matching the test scores of the same student across the two content areas (Object Design and Inheritance). The change in score was calculated by subtracting the student's score obtained with CBA instructional support from the student's score with LNO instructional support. A negative change meant that the student performed better with CBA support versus LNO support.

The results reported in Table 5 indicate no significant differences in the student test scores for either complex questions (Z = 0.000, $p > .05$), simple questions (Z = -0.106, $p > .05$), or the total set of questions (Z = -0.378, $p > .05$). Specifically you should note the Wilcoxon test result (Z = 0.000) for Complex LNO – Complex CBA which indicates that on the complex questions there were just as many students who performed better with lecture notes only (i.e., the LNO treatment) as those who performed better with the CBJava as an instructional support tool (i.e., the CBA treatment). This finding conflicts with the expectation that the CBA treatment should enable a student to perform better on complex questions than with the LNO treatment. One possibility for this finding was that that there may be a carryover effect for those students who received the CBA treatment in period 1 (during the coverage of Object Design) of the study. That is, the CBA treatment may have helped them sufficiently that they performed better than expected on the second assignment for which they did not have the tool available.

Table 5. Test Statistics for Wilcoxon Signed Ranks Test of LNO Treatment vs. CBA Treatment

	Total LNO – Total CBA	Complex LNO – Complex CBA	Simple LNO – Simple CBA
Z	-.378[a]	.000[b]	-.106[a]
Significance (2-tailed)	.706	1.000	.916

[a]Based on negative ranks.

[b]The sum of negative ranks equals the sum of positive ranks.

To answer the carryover effect question a 2 × 2 mixed-design ANOVA was calculated to examine the effects of the group (Group 1 and Group 2) and the treatment (CBA and LNO) on students as indicated by posttest scores. The carryover effect question (i.e., was there a carryover effect of the treatment) was answered once for each set of questions (i.e., Complex and Simple). The 2 × 2 mixed-design ANOVA was used because there is not an alternative non-parametric design that can test for carryover effects. A mixed design ANOVA tests effects of more than one independent variable where at least one of the independent variables must be within-subjects (repeated-measures) and at least one of the independent variables must be between-subjects (Cronk, 2004, p. 74). In the mixed design ANOVA used in this study, the repeated measures variable was treatment (i.e., the ordering of treatments), and the independent variable was posttests.

The effects of the group are essentially the same as a sequence effect because of the ordering of the treatment. Recall that Group 1 was the group of subjects who received the CBA treatment in period 1 (Object Design) and the LNO treatment in period 2 (Inheritance). Group 2 was the group of subjects who received the CBA treatment in period 2 (Inheritance). There was no significant effect indicated by the posttest scores by students on complex questions for the Group × Treatment interaction (F(1,13) = 4.452, *p* > .05). The main effect for treatment on students' posttest scores on complex questions was also not significant (F(1,13) = 0.003, *p* > .05, see Table 6). The main effect for treatment finding is a similar result to the findings of the Wilcoxon Signed Ranks test (i.e., both the non-parametric test and the parametric test came to similar conclusion that there were no treatment effects). However there was a significant effect (i.e., a carryover effect) of treatment on Group for complex questions (F(1,13) = 12.718, *p* < .01, see Table 7). These results indicate that there was a significant carryover effect of Treatment on students as indicated by test scores on complex questions from posttest 1 to posttest 2 for those students who received the CBA treatment in the first period. Students in Group 1 had a similar drop in mean scores (2.14 points) between posttest 1 and posttest 2 on complex questions as did Group 2 (2.25 points) even though it was Group 2 (not Group 1) who received the CBA treatment for the content area (object design) covered by posttest 2 (see Table 3).

Table 6. Tests of Within-Subjects Contrasts on Complex Questions of Group and Treatment

Source	Time	Type III Sum of Squares	*df*	Mean Square	*F*	Sig.	Partial Eta Squared
Treatment	Linear	.021	1	.021	.003	.960	.000
Group × Treatment	Linear	36.021	1	36.021	4.452	.055	.255
Error(Treatment)	Linear	105.179	13	8.091			

Table 7. Tests of Between-Subjects Effects on Complex Questions of Group

Source	Type III Sum of Squares	*df*	Mean Square	*F*	Sig.	Partial Eta Squared
Intercept	11005.952	1	11005.952	783.066	.000	.984
Group	178.752	1	178.752	12.718	.003**	.495
Error	182.714	13	14.055			

$^{**}p < .01$

There was no significant effect indicated by the posttest scores by students on simple questions for the Group × Treatment interaction (F(1,13) = 3.28, p > .05). The main effect for Treatment on simple questions was also not significant (F(1,13) = 0.045, p > .05, see Table 8). The main effect for this treatment finding as indicated by posttest scores on simple questions is a similar result to the findings of the Wilcoxon Signed

Ranks test (i.e., both the non-parametric test and the parametric test came to similar conclusion that there were no treatment effects on students as indicated by posttest scores on simple questions). Finally there was no significant effect on simple questions for Group (F(1,13) = 3.16, p > .05, see Table 9). These results indicate that there was no significant carryover effect on simple questions for those students who received the CBA treatment in the first period.

Table 8. Tests of Within-Subjects Contrasts on Simple Questions of Group and Treatment

Source	Time	Type III Sum of Squares	*df*	Mean Square	*F*	Sig.	Partial Eta Squared
Treatment	Linear	.215	1	.215	.045	.836	.003
Group × Treatment	Linear	15.815	1	15.815	3.282	.093	.202
Error(Treatment)	Linear	62.652	13	4.819			

Table 9. Tests of Between-Subjects Effects on Simple Questions of Group and Treatment

Source	Type III Sum of Squares	*df*	Mean Square	*F*	Sig.	Partial Eta Squared
Intercept	12048.215	1	12048.215	788.449	.000	.984
Group	48.348	1	48.348	3.164	.099	.196
Error	198.652	13	15.281			

The third test performed was a Mann-Whitney *U* test. A Mann-Whitney *U* test was used to compare two independent samples (Group 1 and Group 2) because of the small sample size (Huck, 2004, p. 496). Because the results of the 2 × 2 mixed-design ANOVA test indicated that there was a significant carryover effect of Treatment on students as indicated by test scores on complex questions, this third test was performed for period one only (during which the Object Design content area was covered). That is, the carryover effects confound the Mann-Whitney *U* test for period two (during which the Inheritance content area was covered), therefore it was not performed for period two.

The Mann-Whitney *U* test was calculated examining the treatment effects (CBA versus LNO) on students as indicated by test scores on the various types of test question (Complex and Simple) in period one (Object Design). Primarily this test was utilized to further test the treatment effects on complex questions because this is where the carryover effect occurred; however, for completeness, both simple and the combined set (Simple and Complex) of questions were tested as well.

Students in Group 1 (those who received the CBA treatment first) performed significantly better on the complex questions (*m* rank = 11.50; *U* = 3.500, *p* < .05, *M* =

22.71 vs. *M* = 17.88) than those students in Group 2 (those who received the LNO treatment first). These same students (Group 1) did no better than the students in Group 2 on simple questions (*m* rank = 9.21; *U* = 19.500, *p* > .05, *M* = 22.00 vs. *M* = 19.63). However the students in Group 1 did perform significantly better on the total set of test questions (*m* rank = 11.29; *U* = 5.000, *p* < .01, *M* = 44.71 vs. *M* = 37.50). See Table 10 for the summary of results for this test and Table 3 for the descriptive statistics. The results of the Mann-Whitney U test indicate that the students who were provided with the CBJava tool in the first period of the study (i.e., the CBA treatment) performed significantly better than those students who were only provided with the lecture notes (i.e., the LNO treatment).

Table 10. Mann-Whitney Test - Test Statistics[b] of Group by Question Type for Object Design Content Area

	Complex	Simple	Total
Mann-Whitney *U*	3.500	19.500	5.000
Significance (2-tailed)	.004**	.320	.007**

[b]Grouping Variable: Group (i.e. Group 1 students versus Group 2 students).

***p* < .01

Notes. Complex refers to student performance on complex questions on posttest 1. Simple refers to student performance on simple questions on posttest 1. Total refers to student performance on all questions on posttest 1.

Student Background Assessment Results

A one-tailed Spearman *rho* correlation coefficient was calculated for the relationship between subjects' grade in background coursework (CM111 and PH110) and their test scores for each of the content areas (Object Design and Inheritance) broken out by question type (Simple and Complex). A Spearman *rho* correlation was used because grade is an ordinal measurement. Descriptive statistics for background coursework including means and standard deviations for each group (Group 1 and Group 2) as well as the combined groups are provided in Table 11. The mean scores reported are the average letter grades of the students based on a scale of A = 4, B = 3, C = 2, D = 1, and F = 0.

Although the mean scores for students in Group 1 appear to be better than Group 2, it is only by chance that this occurred. The differences in mean scores does raise a question about the significant effect of the CBA treatment on Group 1 as indicated by scores on posttest 1 found by the Mann-Whitney *U* test. However, the Mann-Whitney *U* test is a ranked-based test, not a means-based test (Huck, 2004, p. 496). It is also a distribution free test (i.e., a normalized distribution is not assumed). Thus, the question about the results of the Mann-Whitney *U* test (i.e., the significant effect of the CBA treatment on Group 1) is mitigated. Still, the differences in student background should be considered when analyzing the total set of results of this study.

Table 11. Descriptive Statistics of Background Coursework by Group

	Group 1			Group 2			Total		
Course	Mean	*N*	*SD*	Mean	*N*	*SD*	Mean	*N*	*SD*
CM111	3.67	6	.516	3.13	8	.835	3.36	14	.745
PH110	2.86	7	1.464	2.38	8	1.188	2.60	15	1.298

Notes. One subject did not take CM111 in Group 1. The mean scores reported are the average letter grades of the students based on a scale of A = 4, B = 3, C = 2, D = 1, and F = 0.

Relevant bivariate correlations for Group 1 are presented in Table 12 and for Group 2 in Table 13. The set of relevant bivariate correlations for the combined groups is provided in Table 14.

Of particular interest was the strong positive correlation found between the background knowledge and the total test scores for both content areas for the combined groups (see Table 14). This finding suggests that students who did well on their background coursework scored well on the posttests, and students who did poorly on their background coursework scored poorly on the posttests. This finding, once again, also raises the question about the significant effect of the CBA treatment on Group 1 as indicated by scores on posttest 1 found by the Mann-Whitney *U* test. Since Group 1 students performed better in their background coursework and because there was a strong positive correlation found between background knowledge and the total test scores, it could be argued that Group 1 students would have most likely performed better on the posttest 1 than Group 2 even without the CBA treatment. Again, this argument can be answered. Notice that for the first period (the period tested by the Mann-Whitney *U* test) there was no significant correlation found for Group 1 between Complex questions and either CM111 (p = .107) or PH110 (p = -.095). This was also the case for Group 2 (i.e., for CM111, p = -.057 and from PH110, p = .463). Recall that the Mann-Whitney *U* test found that students in Group 1 (who received the CBA treatment) performed significantly better than students in Group 2 (who received the LNO treatment) on complex questions but there were no differences found on the simple questions. Thus, even if one argues that Group 1 appears to have had better background knowledge than Group 2, this difference does not appear to have influenced the findings of the Mann-Whitney *U* test.

Table 12. Bivariate Correlations between Background Courses and the Test Scores for each Content Area for Group 1

	Object Design			Inheritance		
Course	Simple	Complex	Total	Simple	Complex	Total
CM111	.853*	.107	.735*	.414	.335	.414
PH110	.667	-.095	.406	.378	.735*	.655

*p < .05

Notes. Bivariate correlations were calculated using the variables posttest score and letter grade. Student posttest scores for each content area broken out by scores on simple, complex, and the total set of questions. The second variable, the students' letter grade, was based on a scale of A = 4, B = 3, C = 2, D = 1, and F = 0.

Table 13. Bivariate Correlations between Background Courses and the Test Scores for each Content Area for Group 2

	Object Design			Inheritance		
Course	Simple	Complex	Total	Simple	Complex	Total
CM111	.139	-.057	.255	.513	-.026	.207
PH110	.199	.463	.281	.168	.444	.279

Table 14. Bivariate Correlations between Background Courses and the Test Scores for each Content Area for Groups 1 and 2 Combined

	Object Design			Inheritance		
Course	Simple	Complex	Total	Simple	Complex	Total
CM111	.497*	.335	.542*	.549*	.330	.519*
PH110	.480*	.429	.477*	.461*	.592*	.561*

*$p < .05$

Results Related to Motivation Questionnaire

A two-tailed Spearman *rho* correlation coefficient was calculated for the relationship between subjects' answers to questions 1-4 and 6-10 of the Motivation Survey (Appendix D) and their test scores for each of the content areas (Object Design and Inheritance) broken out by question type (Simple and Complex). A Spearman *rho* correlation was used because motivation is an ordinal measurement.

In order to address the consistency of the survey questions, the Motivation Survey was developed from similar questions found in other motivation related research in computer science courses (Jenkins, 2001; Mitchell, Sheard, and Markham, 2000; Wilson and Braun, 1985). Additionally, the survey was reviewed and critiqued by an expert in survey development at the study site. By using both the historical and expert sources, consistency in the questions was maintained which attempted to address the reliability and validity issues for the survey (Krathwohl, 1998, p. 435). However, no psychometric analysis of the Motivation Survey was performed.

Descriptive statistics for the Motivation Survey answers including means and standard deviations for each group (Group 1 and Group 2) as well as the combined groups are provided in Table 15. The mean scores reported are averages of the student responses to each question. Responses to each question were based on a four point Likert scale ranging from 1 (extremely important) to 4 (not important). Therefore, low mean scores indicated that the group felt the item was extremely important.

The results of the descriptive statistics show that the combined groups rated questions 1 – 4 (i.e., those questions that were categorized as intrinsic, means ranging from 1.40 – 1.60) more important than questions 6 – 10 (i.e., those questions that were categorized as extrinsic, means ranging from 1.87 to 3.93). A categorized list of each question in abbreviated form is provided in Table 15 for reference. See Appendix D for the complete Motivation Survey.

Table 15. Motivation Survey Closed-Ended Questions Categorized as Intrinsic or Extrinsic

Question Number	Abbreviated Question	Motivation Category
1	enjoy programming	Intrinsic
2	computers are fascinating	Intrinsic
3	subject is intellectually challenging	Intrinsic
4	interested in subject	Intrinsic
6	increase my job security	Extrinsic
7	concerned about computerization of society	Extrinsic
8	help with employment	Extrinsic
9	employer requirement	Extrinsic
10	help with other subjects	Extrinsic

Notes. Question 5 was a social motivation question (Jenkins, 2001) and was inadvertently included in the survey; it was not used in the analysis.

Table 16. Descriptive Statistics of Motivation Survey Questions by Group

	Group 1			Group 2			Total		
Question	Mean	*N*	*SD*	Mean	*N*	*SD*	Mean	*N*	*SD*
1	1.14	7	.378	1.75	8	.886	1.47	15	.743
2	1.29	7	.756	1.63	8	1.061	1.47	15	.915
3	1.29	7	.488	1.88	8	1.126	1.60	15	.910
4	1.00	7	.000	1.75	8	.886	1.40	15	.737
6	1.86	7	1.215	2.00	8	1.069	1.93	15	1.100
7	2.43	7	1.134	2.88	8	.835	2.67	15	.976
8	1.86	7	1.464	1.88	8	1.126	1.87	15	1.246
9	4.00	7	.000	3.87	8	.354	3.93	15	.258
10	2.00	7	1.000	2.13	8	.991	2.07	15	.961

Notes. The means represent the average responses to each closed-ended question of the Motivation Survey. Responses were based on a four point Likert scale: 1 – extremely important, 2 – moderately important, 3 – slightly important and 4 – not important. Questions 1-5 measured intrinsic motivation, and questions 6-10 measured extrinsic motivation. Group 1 refers to those students who received the CBA treatment in period one, Group 2 refers to those students who received the CBA treatment in period 2, and Total refers to both groups combined.

Relevant bivariate correlations for Group 1 and Group 2 are presented in Table 17 and Table 18 respectively. The set of relevant bivariate correlations for the combined groups is provided in Table 19 . A strong negative correlation on questions 1-4 indicates that students who were intrinsically motivated did well on the posttests. A strong negative correlation on questions 6-10 indicates that students who were extrinsically motivated did well on the posttests. Of the fifty-four bivariate correlations calculated each for the Group 1, Group 2, and the combined groups, three were found to be significant for Group 1, none for Group 2, and none for the combined groups. Because the percentage of significant correlations was very small, it could be argued that they occurred by chance. Based on this finding, there were minimal relationships between extrinsic and intrinsic motivation and student performance on posttests.

Table 17. Bivariate Correlations between Motivation Survey Questions and Test Scores by Content Area for Group 1

	Object Design			Inheritance		
Question	Simple	Complex	Total	Simple	Complex	Total
1	-.520	-.624	-.618	-.618	-.535	-.612
2	.000	-.416	-.412	.309	.428	.408
3	.242	-.242	.000	.479	.828*	.791*
4	0	0	0	0	0	0
6	-.030	-.291	-.268	-.169	-.175	-.177
7	-.715	-.476	-.887**	-.330	-.255	-.299
8	-.161	-.242	-.399	.080	-.083	.000
9	0	0	0	0	0	0
10	-.467	-.010	-.171	-.643	-.511	-.558

*$p < .05$. **$p < .01$.

Notes. Bivariate correlations were calculated using variables posttest score and motivation. Student posttest scores for each content area (Object Design and Inheritance) are broken out by scores on simple, complex, and total set of questions. The second variable, motivation, was the average responses to each closed-ended question of the Motivation Survey. Responses were based on a four point Likert scale ranging: 1 – extremely important, 2 – moderately important, 3 – slightly important and 4 – not important.

Table 18. Bivariate Correlations between Motivation Survey Questions and Test Scores by Content Area for Group 2

	Object Design			Inheritance		
Question	Simple	Complex	Total	Simple	Complex	Total
1	.608	.065	.612	.323	.325	.474
2	.097	.028	.223	.532	.049	.099
3	.572	.181	.523	.225	.614	.445
4	.504	.013	.416	.168	.560	.501
6	.415	.051	.519	.604	.203	.333
7	.152	-.351	.121	.215	.064	.039
8	.302	.181	.330	.418	.381	.366
9	.415	-.501	.167	.166	.167	.254
10	.488	-.065	.427	.424	.388	.602

Table 19. Bivariate Correlations between Motivation Survey Questions and Test Scores by Content Area for Combined Groups

	Object Design			Inheritance		
Question	Simple	Complex	Total	Simple	Complex	Total
1	.114	-.483	-.221	-.191	-.145	-.203
2	.044	-.265	-.192	.281	.084	.078
3	.322	-.193	.007	.138	.397	.227
4	.166	-.472	-.220	-.162	.006	-.103
6	.240	-.137	.057	.191	.026	.116
7	-.217	-.438	-.354	-.101	-.113	-.136
8	.088	-.089	-.057	.207	.068	.147
9	.376	-.062	.280	.249	.280	.311
10	.104	-.107	.065	-.012	-.053	.068

*$p < .05$.

Feedback was also collected from one open-ended survey question included in the Motivation Survey. The question was as follows: *I am interested in this course for other reasons not listed. I am listing these here.* In total nine students responded to this question. These responses are provided in Appendix F. Responses were evaluated and assigned to an appropriate thematic category. The number of responses in each category was summed. Finally, each theme was labeled as either intrinsic or extrinsic. Table 20 provides a summary of the results. These results showed that other motivating factors were primarily extrinsic.

Table 20. Motivation Survey Summary of Responses to Open-Ended Question

Theme	Number of Responses	Motivation Category
The course is a requirement of or closely related to the student's degree program.	4	Extrinsic
The student is taking the course because they really enjoy computing and see it as an exciting area to study.	2	Intrinsic
The student wants to have a better understanding of computing for follow-on coursework.	3	Extrinsic

Results Related to Attitudes towards CBJava Questionnaire

A two-tailed Spearman *rho* correlation coefficient was calculated for the relationship between subjects' answers to questions 1-6 of the Student Attitudes towards CBJava Survey (Appendix E) and their test scores for the CBJava supported content area (Object Design or Inheritance) broken out by question type (Simple and Complex). To address the consistency of the closed-ended survey questions (i.e., questions 1-6) two techniques were used. The survey was structured so that any bias towards negative or positive responses was minimized (i.e., questions 1, 2, 4, and 5 were supportive of the tool; questions 3 and 6 were non-supportive, see Table 21). And, the survey questions were developed by the researcher and critiqued by an expert in survey development. However, no psychometric analysis of the survey was performed.

Table 21. Attitudes towards CBJava Closed-Ended Questions Categorized as Supportive or Non-Supportive of CBJava

Question Number	Question	Category
1	CBJava is an effective tool in learning object design/inheritance.	Supportive
2	CBJava helped my understanding of how real world problems can be conceptualized in Java classes.	Supportive
3	I had difficulty relating the examples to the instructional content within CBJava.	Non-Supportive
4	I found that the ability to create concrete examples with expert feedback supported my understanding of the instructional content.	Supportive
5	CBJava supported my understanding of chapter topics by linking the examples to the related topics.	Supportive
6	I found the CBJava tool difficult to use.	Non-Supportive

Recall that Group 1 was the set of students who were supported by the CBJava tool for the Object Design content area (period 1). Group 2 was the set of students who were supported by the CBJava tool for the Inheritance content area (period 2). It was not appropriate to perform correlational tests on content areas for which the groups did not have CBJava support. Correlations were not calculated for the combined groups because of the differences in supported content area coverage. A Spearman *rho* correlation was used because student attitude is an ordinal measurement.

Descriptive statistics for Student Attitudes towards CBJava Survey answers including means and standard deviations for each group (Group 1 and Group 2) as well as the combined groups are provided in Table 22. The mean scores reported are averages of the student responses to each question. Responses to each question were based on a five point Likert scale ranging from 1 – strongly agree to 5 – strongly disagree. Therefore, high mean scores indicated that the group strongly disagreed with the item. The descriptive statistics showed that the combined groups answered more positively to questions (1, 2, 4, and 5) supportive of the CBJava tool (means ranging from 2.07 to 2.33) and more negatively to those questions (3 and 6) that were not supportive of the CBJava tool (means ranging from 3.60 to 3.80). Answering negatively to a non-supportive question indicated that the student disagreed with the non-supportive question. Answering positively to a supportive question indicated that the student agreed with the supportive question. Thus, based upon average responses to questions 1-6 students liked the CBJava tool.

Table 22. Descriptive Statistics of Student Attitudes towards CBJava by Group

	Group 1			Group 2			Total		
Question	Mean	*N*	*SD*	Mean	*N*	*SD*	Mean	*N*	*SD*
1	2.43	7	1.512	2.00	8	.756	2.20	15	1.146
2	2.14	7	1.676	2.38	8	1.061	2.27	15	1.335
3	4.14	7	.690	3.50	8	1.069	3.80	15	.941
4	2.14	7	1.464	2.00	8	.756	2.07	15	1.100
5	2.14	7	1.464	2.50	8	.926	2.33	15	1.175
6	3.71	7	1.380	3.50	8	1.414	3.60	15	1.352

Notes. The means represent the average responses to each closed-ended question of the Student Attitudes towards CBJava Survey. Responses were based on a five point Likert scale: 1 – strongly agree, 2 – agree, 3 – no opinion, 4 – disagree, and 5 – strongly disagree. Group 1 refers to those students who received the CBA treatment in period one, Group 2 refers to those students who received the CBA treatment in period 2, and Total refers to both groups combined. Questions 1, 2, 4, and 5 were supportive of the CBJava tool. Questions 3 and 6 were not supportive of the CBJava tool.

Relevant bivariate correlations for Group 1 and Group 2 are presented in Table 23 and Table 24 respectively. A strong negative correlation for questions 1, 2, 4, and 5 and a strong positive correlation for questions 3 and 6 indicate that students who performed well on the posttests were supportive of the CBJava tool. A strong positive correlation for questions 1, 2, 4, and 5 and a strong negative correlation for questions 3 and 6 indicate that students who performed well on the posttests were not supportive of the CBJava tool. Of the 18 bivariate correlations calculated each for Group 1 and Group 2, 4 were found to be significant for Group 1 and none for Group 2.

Table 23. Bivariate Correlations between Student Attitudes towards CBJava and Test Scores for Object Design Content Area for Group 1

	Object Design		
Question	Simple	Complex	Total
1	-.619	-.600	-.831*
2	-.602	-.532	-.825*
3	.548	.385	.683
4	-.248	-.543	-.548
5	-.629	-.353	-.784*
6	.886**	-.019	.661

**p* < .05, ***p* < .01

Notes. Bivariate correlations were calculated using variables posttest score and student attitudes towards CBJava. Student posttest scores for Object Design were broken out by scores on simple, complex, and total set of questions. The second variable, student attitudes towards CBJava, was the average responses to each closed-ended question of the Student attitudes towards CBJava Survey. Responses were based on a five point Likert scale: 1 – strongly agree, 2 – agree, 3 – no opinion, 4 – disagree, and 5 – strongly disagree. Questions 1, 2, 4, and 5 were supportive of the CBJava tool. Questions 3 and 6 were not supportive of the CBJava tool.

Table 24. Bivariate Correlations between Student Attitudes towards CBJava and Test Scores for Inheritance Content Area for Group 2

	Inheritance		
Question	Simple	Complex	Total
1	-.155	.117	-.237
2	-.041	.494	.169
3	.398	-.394	.079
4	.543	-.117	.079
5	-.516	.431	-.051
6	-.006	-.311	-.025

Feedback was also collected from five open-ended survey questions included in the Student Attitudes towards CBJava Survey. To address the consistency of the open-ended questions the survey questions were structured in a manner where both negative and positive responses were solicited. And, as with the closed-ended questions, the survey questions were developed by the researcher and critiqued by an expert in survey development. However, no psychometric analysis of the survey was performed.

Reponses to each question are provided in Appendix G. Responses were evaluated and assigned to a thematic category. The number of responses in each category was summed. Results of this test showed that the combined group responses were more positive than negative. While the subjects primarily suggested changes to the user interface, they also suggested that the content be adjusted as well, and the majority of the subjects felt that it could be used as an effective learning aid for CM111 – Introduction to Structured Programming. Of most significance was the subjects' positive evaluation of the examples with expert commentary. Below is a list of each open-ended question with a summary of the subjects' responses along with the categorization.

1. Question 1: What did you like about the CBJava tool? Responses were categorized as either positive or negative. Fourteen subjects responded positively about CBJava and one subject responded negatively. Positive responses generally liked the ability to see multiple examples including the expert feedback on the examples submitted.
2. Question 2: What did you dislike about the CBJava tool? Responses were categorized as either user interface improvements or content improvements. Six subjects suggested that the user interface needed improvements, four subjects suggested content improvements, and one subject suggested no changes. User interface improvements generally focused on making the site more graphical. Content improvements focused on providing material that was not already covered in the textbook.
3. Question 3: Did CBJava help you in your understanding of the subject matter? If so, tell us how it helped. If not tell us how it hindered your understanding. Reponses were categorized as either helped, hindered, or neither helped nor hindered in the subjects understanding. Ten subjects responded that CBJava helped in their understanding of the subject matter. Four subjects responded that CBJava neither helped nor hindered their understanding. There were no subjects that responded that CBJava hindered their understanding. These responses

reflected similar responses to the ones made in the open-ended question one of this survey concerning likes of the CBJava tool.

4. Question 4: How do you think CBJava could be improved in the future? Responses were categorized as either user interface improvements or content improvements. Five subjects suggested that the user interface should be improved and six subjects suggested that the content should be improved. These responses were very similar to the ones made in the open-ended question two of this survey concerning dislikes of the CBJava tool.
5. Question 5: Might a tool such as CBJava have helped your understanding in CM111 - Introduction to Structured Programming? If so, tell us how it might help. If not, tell us how it would hinder your understanding? Reponses were categorized as either would have helped or would not have helped in CM111. Nine subjects responded that CBJava would have helped them in CM111 and five subjects responded that CBJava would not have helped them in CM111. Of those subjects who responded favorably, they generally felt that having multiple examples would be a benefit. Those subjects that responded unfavorably generally felt that CM111 requires face-to-face interaction and an instructional aid such as CBJava does not provide that.

Summary

Three research questions were addressed by this study. Tests for the first two research questions involved using two types of non-parametric tests and one parametric test (for carryover effects) at the $p < .05$ level. The third research question was addressed through bivariate correlations, descriptive statistics, and thematic analysis of open-ended questions. Non-parametric tests were used because of the small sample size. The following is a restatement of each research question along with the findings.

1. Is there a statistically significant difference in the performance on simple questions between the Case-Based Reasoning Assisted (CBA) group and the Lecture Notes Only (LNO) group?

   <u>Null Hypothesis</u>: There will be no significant difference in the student performance on the simple questions between the CBA group and the LNO group.

   <u>Alternative Hypothesis</u>: There will be a significant difference in the student performance on the simple questions between the CBA group and the LNO group.

   <u>Findings:</u> The results of the Wilcoxon Signed Ranks test showed no significant difference between the two groups on simple questions. Because of these findings a 2 × 2 mixed-design ANOVA test for carryover effects on simple questions for the CBA treatment was performed. This test found no significant carryover effect on simple questions for the CBA treatment. Based upon the findings no further tests were necessary. However, for completeness a Mann-Whitney *U* test was performed on period one (Object Design). The results of this test also showed no significant difference on simple questions for the CBA treatment in the first period of the study. Therefore, the null hypothesis cannot be rejected.

2. Is there a statistically significant difference in the performance on complex questions between the CBA group and the LNO group?

Null Hypothesis: There will be no significant difference in the student performance on the complex questions between the CBA group and the LNO group.

Alternative Hypothesis: The CBA group will perform significantly better on complex questions than the LNO group.

Findings: The results of the Wilcoxon Signed Ranks test showed no significant difference between the two groups on complex questions. Because of these findings a 2 × 2 mixed-design ANOVA test for carryover effects on complex questions for the CBA treatment was performed. This test found a significant carryover effect on complex questions for the CBA treatment. Based upon the findings of a significant carryover effect a Mann-Whitney *U* test was performed on period one (Object Design). The results of this test showed a significant difference on complex questions for the CBA treatment in the first period of the study, indicating that students who received the CBA treatment performed better than students received the LNO treatment. Therefore, the null hypothesis was rejected.

3. How is student performance on the measures of learner competencies dependent upon motivation, background knowledge, and attitudes towards the case-based hypertext learning tool?

   Findings: Testing for relationships between background knowledge and learner competencies was performed by calculating a one-tailed Spearman rho coefficient. Although there were several significant results, the most interesting result was that a strong positive correlation was found between the background knowledge and the total test scores for both content areas for the combined groups, indicating that students who did well in their background coursework also did well on the posttests.

Testing for relationships between motivation and learner competencies was performed using two methods. A two-tailed Spearman *rho* coefficient was calculated for each of the close-ended questions in relationship to the test scores. Of the 54 correlations calculated each for the Group 1, Group 2, and the combined groups, three were found to be significant for Group 1, none for Group 2, and none for the combined groups. Descriptive statistics show that the combined groups answered more positively towards questions that were categorized as extrinsic and more negatively towards those that were categorized as intrinsic. A thematic analysis was used to analyze the open-ended question. Results of this test showed that other motivating factors were primarily extrinsic.

Testing for relationships between student attitudes towards CBJava and learner competencies was also performed using two methods. A two-tailed Spearman *rho* coefficient was calculated for each of the close-ended questions in relationship to the test scores. Of the 18 correlations calculated each for the Group 1 and Group 2 four were found to be significant for Group 1 and none for the Group 2. Descriptive statistics showed that the combined groups answered more positively towards questions (1, 2, 4, and 5) that were supportive of the CBJava tool and more negatively towards those questions (3 and 6) that were not supportive of the CBJava tool, indicating that students liked the CBJava tool.

A thematic analysis was used to analyze the open-ended questions. Results of this test showed that the combined groups answered more positively towards the use of the tool than negative. The subjects primarily suggested changes to the user interface but also suggested that the content be adjusted as well. Additionally the majority of the subjects felt that it could be used as a effective learning aid for CM111. Of most significance was the subjects' positive evaluation of the examples with expert commentary.

Chapter V: Summary, Discussion, and Recommendations

In this chapter a summary of the study will be presented dealing with a comparison of the results to the literature in the field. Topics in the discussion will include the effects of CBJava and the relationships identified between student performance on the measures of learner competency and motivation, background knowledge, and attitudes towards CBJava. The chapter will conclude with the study's implications and suggestions for future research.

Summary

The primary purpose of this study was to determine if a semi-automated reasoning tool that provides a set of searchable cases (CBJava) would improve a student's understanding of the more difficult concepts in an object-oriented programming course. Specifically, this study tested the learning effects of a semi-automated reasoning tool as an instructional aid in an online object-oriented programming course. Also investigated were the relationships between the dependent variable student performance with independent variables of motivation, background knowledge, and student attitudes towards CBJava. Addressing the use of CBJava as an instructional aid as well as motivation, background knowledge, and attitudes towards CBJava provided direction on how an online course covering a complex topic area might be delivered in the future.

Subjects for the study were randomly assigned to two different orders for receiving the CBJava treatment from two sections of the Object-Oriented Programming 1 course. Object-Oriented Programming 1 is the course offered at an NCAA, Division II university located in the Midwest region of the United States that introduces students to object-oriented programming. Posttests were used to measure the effects of CBJava on learner competency. Background knowledge was collected through student transcripts. Motivation and student attitudes towards CBJava data were collected from surveys. Wilcoxon Signed Ranks test, Mann-Whitney *U* test, mixed-design ANOVA, Spearman *rho* correlations, and thematic analysis were used to analyze the data. All data were collected during the Spring 2005 semester.

Discussion of Research Questions

Effects of CBJava

Two research questions directly examined the effects of the use of CBJava as an instructional support tool on student performance. The questions were similar but investigated different levels of Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956). A discussion of these research questions follows.

Effects of CBJava on Simple Questions. The first research question dealt with student performance on simple questions with and without the use of CBJava. The results of three separate tests, a Wilcoxon Signed Ranks test, a 2 × 2 mixed-design ANOVA test, and a Mann-Whitney U test, found that there was no significant difference between the two groups on simple questions. These findings are consistent with the research by Kozma (1994). His research suggests that effective use of technology is one that is grounded in the "cognitive and social processes by which knowledge is constructed". The simple questions that were tested in this study were ones that fell on the lower end of Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956). Simple questions are typically ones that do not require a lot of deep thought. Quite often the content covered in a face-to-face lecture does not even address these types of questions. Most instructors would make the assumption that a college student should be able to gain the necessary understanding from assigned readings and homework with minimal teacher-learner interaction in order to answer simple questions. Thus, providing a learning aid such as CBJava that is designed to support advanced learning in complex and ill-structured domains would not be expected to provide significant benefit in this area. The findings of my study were consistent with the existing research such as that of Spiro et al. (1992).

Effects of CBJava on Complex Questions. The second research question dealt with student performance on complex questions with and without the use of CBJava. The first test conducted to address this question, the Wilcoxon Signed Ranks test, found no significant difference on complex questions between the two groups. This finding might reasonably be explained by the students' reaction to carryover effects. The possibility of carryover effects was considered because access to the CBJava tool in the first period might have provided the subject with an improved understanding of the content of the subject on the first topic, and thus an a better understanding may have affected or ”carried over” this knowledge into the second period. Thus, a second test, a 2 × 2 mixed-design ANOVA test for carryover effects on complex questions for the CBA treatment was performed.

The 2 × 2 mixed-design ANOVA test for carryover effects did indeed find a significant carryover effect on complex questions for those students who used the CBJava tool in the first period of the study. Thus, a third test was necessary in order to determine why the first test, the Wilcoxon Signed Ranks test, failed to find a significant difference. This third test was conducted on only the first period of the study because the second period was corrupted by the carryover effects. The results of the Mann-Whitney *U* test that compared the CBA group to the LNO group did find a significant difference on complex questions between the two groups on the first period. Based upon the findings of these three tests, the null hypothesis that there would be no significant difference between the two groups on complex questions was rejected.

The results of the Mann-Whitney *U* test showed that the students who, in the first period of the study, were provided with the CBJava tool to support their learning, performed significantly better than those students who were provided with only the lecture notes. The results also showed when considering the entire set of questions (Simple and Complex) the students who used the CBJava tool performed significantly better than those who only had access to the lecture notes, although they did no better on the simple questions. Interpreting these results a bit further, they show that the improvement on the complex questions was of such significance that the improvement drove the overall performance on the entire set of questions.

The complex questions that were tested in this study were questions that fell on the higher end of Bloom's taxonomy of learning objectives (Bloom and Krathwohl, 1956). Complex questions are typically ones that require sustained deep thought. Quite often the content covered in a face-to-face lecture primarily addresses these types of questions either by providing an alternative view of the content or through teacher-learner interaction. Typically complex content does require a significant amount of teacher-learner interaction.

The findings of this study support the alternative hypothesis that providing CBJava as an instructional support tool will improve student performance on complex questions. The findings are also consistent with the Cognitive Flexibility Theory (CFT) described by Spiro et al. (1992). The basic premise of this theory and the related research is that complex subject matter can be learned best if it is provided with multiple views or indexes. CBJava provides this capability.

Background, Motivation, and Attitudes

The focus of the third research question was to understand some of the other factors that might have a relationship with student performance in an online object-oriented programming course. In particular, three relationships were tested by calculating a bivariate Spearman *rho* correlation coefficient. These relationships were: performance and background knowledge, performance and motivation, and performance and student attitudes towards the CBJava tool. Each of these relationships was tested only for the period of the study when all of the students were taking the Object-Oriented Programming 1 course online.

Performance and Student Background. With respect to the performance and student background relationship the most interesting result was the strong positive correlation found between the background knowledge and the total test scores for both content areas for the combined groups. It is also interesting to see that these strong positive correlations were primarily driven by the strong positive correlations found between background knowledge and the test scores on simple questions. That is, each of the four bivariate correlations calculated for the combined groups on performance on simple questions and background knowledge were found to have a strong positive relationship. However, only one of the four bivariate correlations calculated for the combined groups for performance on complex questions and background knowledge were found to have a strong positive relationship (See Table 14).

Background knowledge was measured by the grades received in two prerequisite courses for Object-Oriented Programming 1: Introduction to Structured Programming and Logic for Computer Programming. A strong positive correlation does not indicate causality. It simply means that a relationship exists. However, this finding suggests that students who do well in these prerequisite courses will also do well in an online Object-Oriented Programming 1 course or a similar one. More importantly though it appears that performance in the background coursework cannot be used as a predictor for how the student will handle the complexities of an online object-oriented course. Even so, performance in prerequisite coursework is a factor to be considered before a student takes an online course in object-oriented programming and should be used in advising the student.

Performance and Motivation. The motivation factor was investigated in this study in order to see how students' motivation for taking an object-oriented programming course might affect their performance in an online version of the course. With respect to the performance and motivation relationship the most interesting result was the number of significant correlations. Of the 54 bivariate correlations calculated each for the Group

1, Group 2, and the combined groups, three were found to be significant for Group 1, none for Group 2, and none for the combined groups. Thus, the percentage of significant correlations was very small. It could be argued that just by chance these were found. This finding suggests that the type of motivation be it intrinsic or extrinsic plays a minimal role in how students performed in this online course. Additionally it suggests that motivation type plays no role in determining how a student might perform in an online course in object-oriented programming. It could also reflect something about the questions within the Motivation Survey. Therefore, although the results of this study suggest that motivation need not be a factor that should be considered before a student takes an online object-oriented course, this result is counter intuitive and is an area for further study.

The descriptive statistics show that the combined groups were more intrinsically motivated to take Object-Oriented Programming 1 than extrinsically. In general, the students who took this course really liked computers, felt that the subject was intellectually challenging, and were very interested in the topic. The extrinsically motivating factors were that the course was a requirement for the student's major and that the course would help them get a job.

It is interesting to see that the students who took this course are more intrinsically motivated. This finding is consistent with an overall trend in computer science at the study site. That is, fewer students are majoring in computer science because the perceived job demand has greatly diminished since 2000. Those students who are majoring in computer science are motivated not by money, but rather by interest.

Performance and Attitudes towards CBJava. Student attitudes towards CBJava were investigated in order to see how they might influence the students' performance in an online version of Object-Oriented Programming 1. With respect to the performance and student attitudes relationship the most interesting result was the set of significant relationships that were found. Of the 18 correlations calculated each for the Group 1 and Group 2, four were found to be significant for Group 1 and none for Group 2. Although the number of correlations is small, Group 1 correlations were consistently significant when comparing students' overall performance with respect to their attitudes towards the CBJava tool. The more positive the students in Group 1 felt about CBJava, the better they performed on test questions. It is important to note that Group 1 performed significantly better than Group 2 on performance tests with the aid of CBJava. Several rival explanations exist for this finding.

Since each group reviewed different content areas with the CBJava tool, it could be that the content area accessed by Group 1 was organized better. However, the same organizational structure was used for both content areas, so this explanation is not very convincing. A second explanation could be that the makeup of Group 1 was different that the makeup of Group 2. However, the random assignment mitigates this explanation.

Descriptive statistics showed that the combined groups answered more positively towards questions (1, 2, 4, and 5) that were supportive of the CBJava tool and more negatively towards those questions (3 and 6) that were not supportive of the CBJava tool. Overwhelmingly the students felt that CBJava was an effective instructional support tool.

Results from the thematic analysis of the open-ended questions also support this claim. For example when students were asked about what they liked about the CBJava tool, 14 out of the 15 responses were positive. And, when asked if CBJava helped or hindered their understanding, 10 students responded that CBJava helped and four students responded that in neither helped nor hindered. Of most interest was that subjects really like the examples with expert commentary. Over and over again, the students stated that the examples were really helpful. Students also noted that the expert commentary on the posted examples made the concepts easier to understand through variety.

These results are consistent with the research by Schank and Cleary (1995). Students learn through exploration and experimentation. By exploring both the posted examples as well as reviewing the detailed examples provided by CBJava, the students gained additional insight and understanding of the content. Submitting examples of their own choosing enabled the students to pursue their personal interests as well as providing more breadth in the examples for their peers.

Conclusions

There are several overarching conclusions suggested by the results of this study.

Application of Instructional Design Techniques for Complex Topics

Distance education courses can be developed that are at least as effective as face-to-face courses provided that the proper instructional design techniques are utilized. Improvements in video technology suggest improved methods of lecture content delivery should be considered and tested. At the same time, some particular courses may not be viable candidates for online courses. The pilot study referenced in Chapter 1 raised this as an issue. Some courses contain difficult concepts that may require additional scaffolding for the learner to gain sufficient understanding.

It is possible to design an online course that supports advanced learning of complex and ill-structured content. This requires the proper use of current technology grounded in the teaching architectures such as those listed by Schank and Cleary (1995).

Current technologies such as sofTV provide a simple mechanism for creating hypertext video content allowing the student the ability to navigate the content reviewing the complex material as necessary. This offers a significant advantage over that of a taped lecture. Providing the ability to review and reflect on content is a teaching architecture that supports learning of complex content (Schank and Cleary, 1995). Also, course content management systems such as WebCT relieve the instructor of much of the burden of course management and development. Course artifacts such as lecture notes, sample test questions, and slide presentations can be easily incorporated, managed, and accessed through these systems. Providing a well organized interface to the student is very important in an online course.

Even though we can recreate our face-to-face lectures in a hypertext video format and host our other course artifacts through a course management system, this is still not sufficient for courses that contain complex content. Learning complex content requires additional support over and above that of a lecture based course. This is where the teaching architectures of Schank and Cleary (1995) can be applied. Learning through examples in an online course can be supported simply through a hypertext document. This document needs to provide an alternative set of indexes that allow the student to explore the set of examples which illustrate a particular concept. It reduces the rigidity that lecture oriented courses often have whether they are face-to-face or online. In this way the student can draw out the useful generalizations through their own natural learning.

CBJava as an Instructional Support Tool

This study found that CBJava is an instructional tool that can significantly improve students' understanding of complex subject matter. Through its theoretical grounding in case-based reasoning and Cognitive Flexibility Theory, CBJava provides the additional scaffolding required by students to learn complex and ill-structured concepts. CBJava provides a question and answer interface that allows the student to learn through exploring the content. It avoids the rigidity found in textbooks through its hypertext implementation.

Students found adding their own examples, reviewing examples provided by others, and the expert feedback created an environment that aided them in their learning. Of the positive comments made there appear to be two overarching themes (1) learning through examples is very helpful and (2) CBJava can be generalized to other content areas.

Having examples with expert feedback was very helpful to the student and was a consistent theme throughout the survey comments. As one student put it "I liked how the examples were corrected and resubmitted. This allowed us to see errors and how they were fixed". Providing a mechanism that allows students to submit examples, receive feedback, and see the examples of others supports learning through its constructivist grounding. This study found that students do make significant gains in their understanding of the subject matter when given this type of instruction. These finings are consistent with the learning theory of constructivism and its applications through CFT (Spiro and Jacobson, 1995).

Although expert feedback helps the student, there are costs involved that need to be considered. Expert feedback requires an investment of an instructor's time and talents. In a classroom that contains a large number of students, it may not be feasible to review examples from every student. However, these economic issues can be overcome.

By providing expert feedback on one example and making the original example, its repaired version, and the feedback available to the entire class, all students benefit. In a large classroom, a random set of submitted examples can be reviewed and repaired rather than the entire set. This would alleviate some of time burden on the instructor. Additionally the classroom size could be limited. In fact at the study site, the online class size limit is typically 24 students. Another method might be to have former students provide the expert review.

It is reasonable to require online instructors to provide feedback to students. Traditionally this feedback comes in the form of emails, discussion threads, graded assignments, and graded tests. Providing expert review of posted examples is another alternative to this set of instructor-learner interactions from which the instructor can choose from in order to support student learning.

CBJava can be generalized to many different content areas, and it is not simply limited to object-oriented programming. Its underlying relational architecture is easily extendable to other domains with minimum technical support from a programmer. Additionally, as part of this study students were surveyed about how a tool such as

CBJava might have helped them in Introduction to Structured Programming, a prerequisite course to Object-Oriented Programming 1. Generally the students felt that having a similar tool would be beneficial. However what is most interesting in their comments were the negative components. Some of the students felt that the content of CM111 was too simple and that having a tool such as CBJava would offer no benefit to these simple topics. This is quite a perceptive finding by the students. Consistent with the findings of this study, CBJava would offer no significant advantage to learning simple content. Thus, although CBJava offers a framework that may be generalized to other courses, it is only the complex content that should be addressed by such a tool.

Not all students found CBJava to be beneficial. Of the negative comments made there also appear to be two overarching themes (1) the user interface was too plain and (2) there was too much repetition. Of these themes, the first one is not significant. Although more color and graphics could be added to the interface, it is doubtful that it would have contributed to their learning.

The second theme, too much repetition, is consistent with the literature. The Cognitive Flexibility Theory requires that multiple indexes be provided into the same set of content (Spiro, et al., 1988). CBJava provides examples with multiple links to different components of the same example as well as links to multiple examples that illustrate the same concept. Some learners may find this difficult to use and understand, and this was found to be true in this study. However, this study found that although these students may have perceived this to be a distraction, in fact this "distraction" improved their learning.

Background and Motivation Factors for Advising

When advising a student to take an online course (1) the student's performance in prerequisite courses should be considered and (2) the student's type of motivation, that is, intrinsic or extrinsic, need not be considered although perhaps other aspects of motivation might be considered such as social or achievement (Jenkins, 2001). Findings of this study showed a significant relationship between the students' performance on all types of questions in the online object-oriented course and how they performed in the prerequisite courses. Thus, this consideration should be made regardless of whether the content of the online course is simple or complex. This finding also implies that if a student has done poorly in the previous related coursework, they will probably do poorly in the online course. Conversely, if a student has done well in the prerequisite coursework, they will likely do well in the online coursework. Therefore, performance in background coursework should be considered by both the advisor and the student before the student takes the online course.

Results of this study found no consistent relationship between the type of motivation of students and how they performed in the online course. Students tended to be more intrinsically motivated to take Object-Oriented Programming 1 which is consistent with the trends in computer science at the study site, that there is an overall decline in the enrollment numbers in computer science courses at the study site; the ones that are enrolling are doing so because they generally like the discipline. However, an intrinsic motivation has no relationship with how a student performs in an online course.

Recommendations for Practice and Future Research

The results of this study lead to the following recommendations for practice and suggestions for future research.

Recommendations for Practice

It is important that educators take advantage of course management systems when implementing the learner-content course interaction in order to mitigate the economic development costs. Developing online courses should not be an all-encompassing endeavor. Online components can be easily created by using the existing instructional artifacts that have been developed and proven effective over time by the instructor. Face-to-face lectures can be recorded using software such as sofTV. SofTV is a technology that creates hypertext videos which are a significant improvement over video tapes. Other components such as lecture notes, sample test questions, and homework assignments can be easily integrated into an online course using a course management system such as WebCT.

Supporting complex content through examples that illustrate both good and bad alternatives is an important method of scaffolding students' understanding. Learning complex topics can be supported through examples where the learner can construct their own knowledge. Complex topics covered in an object-oriented programming course should be illustrated with both correct and incorrect examples. Both types of examples should have expert commentary tied to them with reasons why they are incorrect or correct.

Student advising should emphasize an assessment of student performance in background coursework in order to guide the student in choosing an online object-oriented course. Incorporating this type of assessment into a questionnaire for students who are considering coursework that is online would be beneficial for both the student and the advisor. Although there are many factors that should be considered before taking an online course, poor performance in prerequisite courses is an indicator that the student will have difficulties with the follow-on course. Because of the other course interaction issues that arise in an online course, both the advisor and the student should strongly consider performance in prerequisite coursework before enrolling in the online course.

Complex content requires enhanced instructor-learner interactions. These types of interactions are very difficult to replicate in an online course. Providing instructional support for complex content is important regardless of whether or not the course is online or face-to-face. However it is most problematic in the online course. An instructional support tool such as CBJava is therefore a necessary component of an online course. Alternatives such as threaded discussions may work as well, but do not allow for the natural learning that case-based tool such as CBJava provides.

Economic costs to setup a course supported with a case-based tool based upon CBJava's framework need to be considered. Currently CBJava's framework is extendable to other courses, but its extension currently requires technical support from a programmer. Development of the web pages with multiple indexing schemes similar to those in CBJava does require a significant amount of the instructors' time and talents. Therefore, it might be more feasible to employ an instructional designer to assist with the initial setup of the tool rather than expecting the instructor to develop it themselves.

Recommendations for Future Research

Further research should be conducted to determine relationships between student performance in an online course and the related prerequisite courses. This investigation is a more generalized form of what this study found to be true for Object-Oriented Programming 1. This data could be collected from existing student records.

The effects of CBJava on student performance on both simple and complex questions in a face-to-face object-oriented course should be investigated. This study found that when students used CBJava as an instructional support tool for an online class they performed significantly better on the complex assessment questions than those who did not. It would be interesting to see if this effect would be similar for that of a face-to-face course.

CBJava's functionality could be extended to incorporate other complex content areas related to object-oriented programming in Java. Initially these areas should be those that are necessary in the first course in object-oriented programming such as the ones covered in Object-Oriented Programming 1 such as interfaces, polymorphism, user interface design, recursion, event handling, and exception handling.

Providing one or more overarching application examples within CBJava would be helpful to the student. A complete example that is fully elaborated and includes components from all object-oriented content areas could support the breadth of the course much like a case study. This example could potentially cross multiple computer science courses as well. Students often complain about not being able to see the big picture, and this type of overarching example might mitigate this complaint. Although the architecture within CBJava supports this capability, an example like this has not been fully integrated.

Research on how the framework of CBJava can be applied to other object-oriented programming courses in other languages is another viable area to explore. For example the content area of inheritance is applicable to all object-oriented programming languages. Examples can be illustrated in any object-oriented language. It would be quite interesting to see the same example implemented in Java, C++, and C#. Additionally, the CBJava framework could be used as a basis to create other case-based assisted tools such as CBC++ (Case-based C++) and CBC# (Case-based C#).

Enhancements to CBJava's framework should be considered as well. How might the framework be made more extendable? Could the framework be developed so that it could be extended to any course without the support of a programmer? Significant opportunities lie in providing an easily extendable framework that would allow any instructor to setup their own case-based assisted content areas with minimal technical assistance.

Finally, research opportunities exist to determine how the framework of CBJava can be applied to other computer science courses as well as other disciplines. CBJava can easily be extended to support advanced programming courses as well as other computer science courses. For example advanced programming topics such as data structures, multithreading, and networking could be implemented as content areas within CBJava.

The CBJava framework could be extended to support other computer science courses such as database design or software engineering. For example one of the more complex topics in database design is the development of the entity relationship model of a problem domain. This task is complex because more than one entity relationship model can represent the same problem domain, and each of these models has their own positive and negative attributes. Therefore, having a tool similar to CBJava to provide multiple models of the same problem domain would be very effective. It would help students understand the complexities of this content area.

Finally, the CBJava framework could be applied to other disciplines. For example, the learning of algebra could be supported with this framework. Conceptually difficult problems in mathematics such as word problems can be decomposed into simpler components and reassembled in order to solve the problem. It is this decomposition and reassembling through examples that makes CBJava an effective instructional support tool for such a course. Other applications of the CBJava framework are numerous and are limited only by one's imagination.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications, 7*(1), 39-59.

Biddle, R., & Tempero, E. (1998). Java pitfalls for beginners. *SIGCSE Bulletin* (Association for Computing Machinery, Special Interest Group on Computer Science Education*), 30*(2), 48-52.

Bloom, B. S., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives; the classification of educational goals, by a committee of college and university examiners* (1st ed.). New York: Longmans, Green.

Boster, F. J., Meyer, G. S., Roberto, A. J., & Inge, C. C. (2002). *A report on the effect of the unitedstreaming™ application on educational performance*. Farmville, VA: Longwood University.

Bruner, J. (1996). *The culture of education*. Cambridge, MA: Harvard University Press.

Bruno, M. S. (2000). *The case of the older shoulder: A resource for case-based teaching.* Retrieved April 28, 2003 from http://carbon.hampshire.edu/~mbruno/ns121/

Clark, R. E. (1994). Media will never influence learning. *Educational Technology Research and Development, 42*(2), 21-29.

Collins, A., Brown, J.S. & Holum, A. (1991). Cognitive apprenticeship: Making things visible. *American Education, 15*(3), 38-46.

Collins, M. P., & Berge, Z. L. (1996). Mailing lists as a venue for adult learning. *Eastern Adult, Continuing and Distance Education Research Conference,* The Pennsylvania State University, University Park, PA.

Cronk, B. C. (2004). How to use SPSS$^{©}$ (3rd ed.). Glendale, CA: Pyrczak Publishing.

Deitel, H. M., & Deitel, P. J. (2003). *Java how to program* (5th ed.). Upper Saddle River, NJ: Prentice Hall.

Dewey, J. (1910). What is thought? *How we think* (pp. 1-13). Boston: D.C. Heath & Co., 1-13

Edelson, D. (1998). In R. C. Schank (Ed.), *Inside multi-media case based instruction* (pp. 103-174). Mahwah, NJ: Erlbaum.

Guzdial, M. (2001). Use of collaborative multimedia in computer science classes. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education.* Canterbury, United Kingdom, 17-20.

Hillman, D. C., Willis, D. J., & Gunawardena, C. N. (1994). Learner-interface interaction in distance education: An extension of contemporary models and strategies for practitioners. *The American Journal of Distance Education, 8*(2), 30-42.

Huck, S. W. (2004). *Reading statistics and research* (4th ed.). Boston: Pearson.

Jenkins, T. (2001). The motivation of students of programming. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education.* Canterbury, United Kingdom, 53-56.

Jensen, K., & Wirth, N. (1991). *Pascal user manual and report* (3rd ed.). New York: Springer Verlag.

Keppel, G., Saufley, William H. Jr., & Tokunaga, H. (2003). *Introduction to design & analysis: A student's handbook* (2nd ed.). New York: W.H. Freeman and Company.

Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice-Hall.

Kolling, M. (2004). *BlueJ - the interactive java environment.* Retrieved June 30, 2004 from http://www.bluej.org

Kolling, M. (1999). The problem of teaching object-oriented programming part II: Environments. *Journal of Object-Oriented Programming, 11*(9), 6-12.

Kolling, M., Quig, B., & Patterson, A. (2003). The BlueJ system and its pedagogy. *Computer Science Education, 13*(4), 249-268.

Kolodner, J. (1993). *Case-based reasoning*. San Francisco, CA: Morgan Kaufmann.

Kolodner, J. L., & Guzdial, M. (2000). Theory and practice of case-based learning aids. In D. H. Jonassen, & S. M. Land (Eds.), *Theoretical foundations of learning environment* (pp. 215-242). Mahwah, NJ: Lawrence Erlbaum Associates.

Kozma, R. B. (1994). Will media influence learning? Reframing the debate. *Educational Technology Research & Development* 42(2), 7-19.

Krathwohl, D. R. (1998). *Methods of educational and social science research: An integrated approach* (2nd ed.). New York: Longman.

Leake, D. (1996). CBR in context: The present and future. In D. Leake (Ed.), *Case-based reasoning: Experiences, lessons and future directions* (pp. 3-30). Menlo Park, CA: AAAI Press/MIT Press.

Lepper, M. R. (1988). Motivational considerations in the study of instruction. *Cognition and Instruction, 5*(4), 289-309.

Lindsey, S. D. (2003). On-demand lectures create an effective distributed education experience. *T.H.E. Journal, 31*(4), 16ff.

Lynch, T. (2002). LSU expands distance learning program through online learning solution. *T.H.E. Journal, 29*(6), 47-48.

Madden, M., & Chambers, D. (2002). Evaluation of student attitudes to learning the java language. *Proceedings of the Inaugural Conference on the Principles and Practice of Programming, 2002 and Proceedings of the Second Workshop on Intermediate Representation Engineering for Virtual Machines, 2002,* Dublin, Ireland, 125-130.

March, T., & Ozline.com. *Why WebQuests?, An introduction.* Retrieved June 26, 2004 from http://www.ozline.com/webquests/intro.html

McIsaac, M. S., & Gunawardena, C. N. (1996). Distance education. In D. H. Jonassen (Ed.), *Handbook of research for educational communications and technology: A project of the association for educational communications and technology* (pp. 403-437). New York: Simon & Schuster Macmillan.

Merriam-Webster Inc. (2004). *Merriam-Webster online dictionary.* Retrieved June 24, 2004 from http://www.m-w.com/

Meyer, K. A. (2002). Quality in distance learning. *ASHE-ERIC Higher Education Reports, 29*(4), 1-121.

Meyer, K. A., & ERIC Clearinghouse on Higher Education, Washington, DC. (2002). *Quality in distance education.* (EDO-HE-2002-09 ed.). U.S.; DC:U.S. Govt.

Mitchell, M., Sheard, J., & Markham, S. (2000). Student motivation and positive impressions of computing subjects. *Proceedings of the Australasian Conference on Computing Education,* Melbourne, Australia, 189-194.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Moore, M. G. (1989). Editorial: Three types of interaction. *The American Journal of Distance Education, 3*(2), 1-7.

Mulford, D. (2001). DIY (or do it yourself) streaming video: A constructive approach. *College & University Media Review, 8*(1), 35-56.

Nelson, T. H. (1990). *Literary machines* (90.1st ed.). Sausalito, CA: Mindful Press.

Nelson, W. A. (1994). Efforts to improve computer-based instruction: The role of knowledge representation and knowledge construction in hypermedia systems. *Computers in the Schools, 10*(3/4), 371-400.

Perkins, D.N. (1999). The many faces of constructivism. *Educatonal Leadership, 57*(3), 6-11.

Raab, J., Rasala, R., & Proulx, V. K. (2000). Pedagogical power tools for teaching java. *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education,* Helsinki, Finland, 156-159.

Reed, R. (2003). Streaming technology improves student achievement. *T.H.E.Journal, 30*(7), 14ff.

Roberts, E. (2004). The dream of a common language: The search for simplicity and stability in computer science education. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education,* Norfolk, Virginia, USA, 115-119.

Russell, T. L. (1999). *The no significant difference phenomenon*. Raleigh, NC: North Carolina State University.

Schank, R. C. (1996). Goal-based scenarios: Case-based reasoning meets learning by doing. In D. Leake (Ed.), *Case-based reasoning: Experiences, lessons, and future directions*. (pp. 295-348) Menlo Park, CA: AAAI Press/MIT Press.

Schank, R. C. (1990). *Tell me a story: A new look at real and artificial memory*. NewYork: Charles Schreibner's Sons.

Schank, R. (1982). *Dynamic memory: A theory of learning in computers and people*. Cambridge, England: Cambridge University Press.

Schank, R. C., & Cleary, C. (1995). *Engines for education.* Retrieved June 10, 2004 from http://www.engines4ed.org/hyperbook/

Schmidt, C. P. (2004). *CBJava.* Retrieved July 17, 2004 from http://192.104.1.46/cbr/cbrJava/index.jsp. Unpublished work.

Schmidt, C. P. (2002, Fall). A survey of currently implemented webquests in computer science and computer technology. *Journal of Educational Computing, Design, & Online Learning, Vol. 3*.

Schmidt, C. P., & Lalonde, D. (2004). *CBRubric.* Retrieved June 20, 2004 from http://192.104.1.46/cbr/index.html. Unpublished work.

Sfard, A. (1998). On two metaphors for learning and the dangers of choosing just one. *Educational Researcher, 27*(2), 4-13.

Smulders, D. (2004). *Course development costs.* Retrieved Spring 2004, from http://www.docntrain.com/libdnld/smulder.html.

Spencer, D. D. (1993). *Computer dictionary* (4th ed.). Ormond Beach, Fla.: Camelot Pub. Co.

Spiro, R. J., Coulson, R. L., Feltovich, P. J., & Anderson, D. K. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. (Technical Report No. 441). Champaign, IL: University of Illinois at Urbana-Champaign.

Spiro, R. J., Feltovich, R. P., Jacobson, M. J., & Coulson, R. L. (1992). Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In T. M. Duffy, & D. H. Jonassen

(Eds.), *Constructivism and the technology of instruction: A conversation* (pp. 57-76). Hillsdale, NJ: Erlbaum.

Spiro, R. J., & Jacobson, M. J. (1995). Cognitive flexibility, and the transfer of complex knowledge: An empirical investigation. *Journal of Educational Computing Research, 12*(4), 301-333.

Stanfill, C., & Waltz, D. L. (1988). The memory-based reasoning paradigm. *Proceedings of the DARPA Case-Based Reasoning Workshop,* Clearwater Beach, FL, 414-424.

Webopedia. (2004). *Webopedia.* Retrieved August, 2004, from http://www.webopedia.com

Williams, R., & Cummings, S. (1993). *Jargon :An informal dictionary of computer terms*. Berkeley, CA: Peachpit Press.

Wilson, B.G. & Myers, K. M. (2000). Situated cognition in theoretical and practical context. In D.H. Jonnasen & S.M. Land (Eds.). *Theoretical foundations of learning environments*, pp. 57-88. Mahweh, NJ: Erlbaum.

Wilson, J. D., & Braun, G. F. (1985). Psychological differences in university computer student populations. *Proceedings of the Sixteenth SIGCSE Technical Symposium on Computer Science Education,* New Orleans, Louisiana, United States, 166-177.

Yang, D., & Wei, S. (1999). A project-based approach to teaching introductory computer science. *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference,* San Juan, Puerto Rico, 11b3/19 – 11b3/21.

Appendix A: Kansas State University Consent Form

KANSAS STATE UNIVERSITY

INFORMED CONSENT TEMPLATE

PROJECT TITLE: **Cognitively Flexible Hypertext in an Object Oriented Programming Course: Effects of Case-Based Instructional Support on Student Learning**

APPROVAL DATE OF PROJECT: _____ **EXPIRATION DATE OF PROJECT:** **May/2005**

PRINCIPAL INVESTIGATOR: CO-INVESTIGATOR(S): **Dr. Diane McGrath, Cecil Schmidt**

CONTACT AND PHONE FOR ANY PROBLEMS/QUESTIONS: **785-231-1010 ext. 1161**

IRB CHAIR CONTACT/PHONE INFORMATION: **Rich Scheidt, Chair, Committee on Research Involving Human Subjects, 1 Fairchild Hall, Kansas State University, Manhattan, KS 66506, (785) 532-3224.**
Jerry Jaax, Associate Vice Provost for Research Compliance and University Veterninarian, 1 Fairchild Hall, Kansas State University, Manhattan, KS 66506, (785) 532-3224

SPONSOR OF PROJECT: **Kansas State University**

PURPOSE OF THE RESEARCH: **The purpose of this study is to determine if a case-based hypertext tool can be used to learn complex content in an online object-oriented programming course.**

PROCEDURES OR METHODS TO BE USED: **Students will be asked to use a hypertext learning tool to support their learning of a complex topic in the Object Oriented Programming 1 course. Assessment questions covering the complex topics will be administered that are similar to the types of assessment questions normally given over the content. Additionally students will be asked to fill out two surveys that will provide information related to motivation as well as their attitudes towards the case-based hypertext tool. Background knowledge will be collected based upon the student's university transcript or by a simple interview.**

ALTERNATIVE PROCEDURES OR TREATMENTS, IF ANY, THAT MIGHT BE ADVANTAGEOUS TO SUBJECT:

In the case that there is a significant difference between the groups who had access to the hypertext learning tool and those that did not, safeguards have been built into this study. These procedures are documented in the "RISKS ANTICIPATED" section below. It should also be noted that the course will have the same content and delivery as normal in addition to having the new hypertext learning tool.

LENGTH OF STUDY: **3 Weeks**

RISKS ANTICIPATED: **Because each student will be tested over two content areas, one with the new hypertext learning tool and one without there is risk involved. In particular**

there is a risk that students who use the tool may perform better on the assessment tests than students who did not use the tool. Therefore safeguards have been put into place that will mitigate this potential for harm. In particular if it is determined that there is a significant difference in student understanding based on test scores between the two groups, then the group with the lower scores will have their scores weighted appropriately. Additionally, at the end of the study all students will be given access to the tool for both content areas. And, finally a safeguard has been built into the course schedule that provides additional time to cover these two content areas if necessary. Again, at that time the tool will be provided to both groups for both content areas.

BENEFITS ANTICIPATED: **improved learning of the content**

EXTENT OF CONFIDENTIALITY: **All quantitative data will be reported in aggregate form with no linking to any particular individual's name. All survey and background data will be reported anonymously as well. All data will be maintained in a secure location for the period of three years which is the normal duration for this type of study.**

IS COMPENSATION OR MEDICAL TREATMENT AVAILABLE IF INJURY OCCURS: **n.a.**

PARENTAL APPROVAL FOR MINORS: **No minors will be allowed to participate in the study.**

TERMS OF PARTICIPATION: I understand this project is research, and that my participation is completely voluntary. I also understand that if I decide to participate in this study, I may withdraw my consent at any time, and stop participating at any time without explanation, penalty, or loss of benefits, or academic standing to which I may otherwise be entitled.

I verify that my signature below indicates that I have read and understand this consent form, and willingly agree to participate in this study under the terms described, and that my signature acknowledges that I have received a signed and dated copy of this consent form.

(Remember that it is a requirement for the P.I. to maintain a signed and dated copy of the same consent form signed and kept by the participant

Participant Name: ______________________________

Participant Signature: ______________________________ **Date:** ________________

Witness to Signature: (project staff) ________________________ **Date:** ________________

Appendix B: Object Design Quiz

Name: _________________________ OOP-1 Quiz: Object Design Schmidt

True/False - Place a T for True or an F for false in the space provided next to the question. (12 points)

1. _____ It is recommended that the return type of a mutator method be void.
2. _____ Static variables are also called instance variables.
3. _____ If you do not import a class, you can still access it by specifying its package prefix.
4. _____ In Java, static constants are often declared public.
5. _____ Packages may contain more than one class.
6. _____ Using static variables is considered a good practice in a Java program.
7. _____ In Java, string objects are immutable.
8. _____ Local variables are accessible outside the block in which they are defined.
9. _____ All non-void methods must have at least one return statement in them.
10. _____ It is generally best to have a single method inquire about the state of an object or change the state of the object but not both.
11. _____ It is good practice to maximize the coupling (i.e. dependency) between classes.
12. _____ If an error occurs in a method of an object, the method should always print out an error message.

Multiple Choice – Circle the letter next to the most correct answer. (14 points)

1. We have a class Book that holds objects of type Page. We say that
   A) Page depends on Book.
   B) Book depends on Page.
   C) there is no dependency between the two classes.

2. When a method changes anything other than the state of its implicit parameter it is said to have a(n):
   A) modifier
   B) static method
   C) accessor method
   D) side effect

3. A method that does nothing other than return the value of an instance variable is an example of:
   A) a side effect.
   B) good programming practice.
   C) a mutator method.
   D) an accessor method.

4. Preconditions should be specified:
   A) in comments at the beginning of the method.
   B) in the method header
   C) in the return statement
   D) in the class body

5. If a method is called in violation of a precondition, which of the following are good choices for the action to be taken?
   A) Reformat the hard drive.
   B) Throw an exception.
   C) Perform the calculation assuming precondition is true.
   D) Print a warning message to System.out.
   E) Both B and C

6. in the code segment:

```
public void set(double unitCost, double area)
{
...
   price = unitCost * area;
}
```

the unqualified variable name price means

A) area.price
B) set.price
C) Class.price
D) this.price

7. Given the following method signature, what is/are its explicit parameter(s)?

```
public int myMethod (double val1, int val2)
```

A) this, double, int
B) val1, val2
C) this, val1, val2
D) parameter 0, parameter 1

Short Answer

1. How might you redesign the class below so that it would be more cohesive? Answer this question by providing the new class definition or set of classes. Do not include methods. (4 points)

```
public class  Car {
        private String make;
        private String manufacturer;
        private String engineModelName;
        private String engineModelHorsePower;
        private String engineModelCost;

        //methods
        …
}
```

2. Identify the dependencies between each of the follow sets of classes. (e.g. Student depends on Advisor) (4 points):

   a) Cup, CupHolder

   b) Person, Address

   c) Engine, Car

   d) PC, HardDrive

3. Given the following class definition. Redesign the class so that it has methods that do one and only one thing, i.e. they either access or mutate the object. (4 points)

```
public class Account{
        private double balance;
        public double deposit( double amount) {
                balance = balance – amount;
                return balance;
        }
}
```

4. Consider the problem description below identify the classes that you would use to implement the problem. (4 points)

   Students declare a major in a degree. The degree is assigned and administered by a department. Each student is assigned an advisor.

5. Provide the Java implementation for the following description of a stock tank. Do not be concerned about documenting the pre or post conditions, but provide the necessary checks to ensure that a tank is never overfilled or below empty.

   A stock tank has a water level that is measured in integer increments. You can add water to the tank or drain water from the tank. The tank has a maximum water level of 36 inches and a minimum water level of 0. (4 points)

6. What are the preconditions of the following methods? (4 points)

```
public void setName(String aName) {
        if (aName.length() > 30)
                throw new IllegalArgumentException("Name is too long");
        if (aName.length() = 0)
                throw new IllegalArgumentException("Name is undefined.");
        name = aName;
}
```

```
public void setDayOfWeek(int day)
        if (day > 5 || day < 1)
                throw new IllegalArgumentException("Invalid day");
        dayOfWeek = day;
}
```

Appendix C: Inheritance Quiz

Name: _________________________ OOP-1 Quiz: Inheritance Schmidt

True/False - Place a T for True or an F for false in the space provided next to the question. (12 points)

1. _____ The superclass inherits behavior (methods) and state (attributes) from the subclass.
2. _____ A superclass may only have methods, i.e. behavior.
3. _____ You should minimize the use of inheritance. Whenever possible you should write your own classes rather than extend existing ones.
4. _____ When designing class hierarchies, the common features are placed in the superclass.
5. _____ A subclass can access all public and private fields of its superclass.
6. _____ When used, the call to the super() must be placed as the first statement of the subclass constructor.
7. _____ You should define the equals method to test whether two objects have equal state.
8. _____ Object is the superclass of all classes.
9. _____ It is a good practice to shadow all instance variables of the superclass.
10. _____ Java supports multiple inheritance.
11. _____ An important reason for using inheritance is code reuse.
12. _____ Given that the Employee class extends Person class and this segment of code:

    ```
    Employee boss = new Employee();
    Person terry;
    ```

    The following assignment is valid**: terry = boss;**

Multiple Choice (14 points) – Circle the letter next to the most correct answer.

1. Which of the following statements will make Student a subclass of the Person class?
   A) public class Student implements Person
   B) public class Person implements Student
   C) public class Student extends Person
   D) public class Person extends Student
   E) public class Student subclasses Person

2. Consider these class definitions:

   public class FirstClass { ... }
   public class SecondClass extends FirstClass{ ... }
   public class ThirdClass extends SecondClass { ... }

   Which of the following statements is true?

   A) ThirdClass is invalid. It can not extend a class that itself extends a class.
   B) These classes are valid but ThirdClass can not be extended.
   C) These are all valid class definitions.

3. When defining methods of a subclass, which of the following are true?
   A) You can inherit methods from the superclass.
   B) You can define new methods in the subclass.
   C) You can override methods of the superclass.
   D) All of the Above

4. When defining instance fields for a subclass, which of the following are true?
   A) You can override fields from the superclass.
   B) You can inherit fields from the superclass
   C) You can define new fields.
   D) Both B and C

5. To call the constructor of a superclass, use:
   A) super()
   B) class()
   C) superclass()
   D) upper()

6. If you do not supply an access control modifier, then the default is
   A) protected
   B) package access
   C) private
   D) public

7. Assume these two class definitions:

```
public class Adder
{
   public void addIt(double amount)
   {
     balance = balance+amount;
   }
   public double getBalance()
   {
     return balance;
   }
private double balance;
}
```

And:

```
public class NewAdder extends Adder
{
   public void addIt(double amount)
   {
   balance = balance + amount;
   count = count +1;
   }
private double balance;
private int count;
}
```

Now consider code segment:

```
NewAdder myAdd = new NewAdder();
myAdd.addIt(500);
System.out.println(myAdd.getBalance());
```

What will be printed:

A) 250.0
B) 500.0
C) nothing
D) 0.0

Short Answer (24 points)

1. In the following pairs of classes, identify the top level superclass and the subclasses by circling the top level superclass. (4 points)

   - Employee, Manager, Person
   - Square, Polygon, Triangle,
   - Vehicle, Car, Truck
   - BankAccount, CheckingAccount, Account

2. Draw an inheritance diagram that shows the inheritance relationships between the classes: (4 points)

   Person, Employee, Student, Instructor, Classroom, Object, Lab, Room

3. How might you redesign the class below so that it would make better use of inheritance (i.e. refactor Student into one or more classes)? Answer this question by providing the new class definitions. Do not include methods (4 points).

```
public class Student {
        private String studentId;
        private String undergradAdvisor;
        private String studentName;
        private Date gradSchoolAcceptedDate;
        private String gradResearchDirector;
}
```

4. Write the following class definitions in Java using inheritance that implement the problem description below. In your class definitions only show the attributes. Do not be concerned with the methods. Note: the problem is in relationship to an inventory tracking process. (4 points)

   Each part has a part number and a description. Each part can be categorized as a make part (built in house) or a buy part (purchased). Buy parts have a purchase price. Make parts have an engineering drawing number assigned to them.

5. Given the implementation in **Figure 1** below show the implementation of a savings account for a preferred customer. A preferred customer savings account always receives an additional 10 dollars in interest over and above the normal interest. (4 points).

6. Given the implementation in **Figure 1**. Write the Java implementation for the toString() method in the BankAccount and SavingsAccount classes using inheritance. Recall that this.getClass().getName() returns the class name of an object. (4 points)

   Provide the implementation for the toString() method of the BankAccount class here:

   Provide the implementation for the toString() method of the SavingsAccount class here:

Figure 1

```
public class BankAccount
 {
   public BankAccount(){
      balance = 0;
   }

   public BankAccount(double initialBalance){
      balance = initialBalance;
   }

   public void deposit(double amount) {
      double newBalance = balance + amount;
      balance = newBalance;
   }

   public void withdraw(double amount) {
      double newBalance = balance - amount;
      balance = newBalance;
   }

   public double getBalance() {
      return balance;
   }

   private double balance;
 }

 public class SavingsAccount extends BankAccount
 {
   public SavingsAccount(double rate) {
      interestRate = rate;
   }

   public void addInterest()  {
      double interest = getBalance() * interestRate / 100;
      deposit(interest);
   }

   private double interestRate;
 }
```

Appendix D: Motivation Survey

Motivation Questionnaire

How important is each of the following reasons in your decision to take Object-Oriented Programming 1 course. In each case mark

A) if it is **extremely** important.
B) if it is **moderately** important.
C) if it is only **slightly** important.
D) if it is **not** important at all

1. I enjoy programming. _____
2. I find computers fascinating. _____
3. I find the subject intellectually challenging. _____
4. It is a subject that I am interested in. _____
5. It was recommended by other students or by my family. _____
6. I expect Computer Sciences coursework to increase my job security. _____
7. I am concerned about the computerization of society. _____
8. It will be useful in helping me get a job or stay employed. _____
9. It was a requirement of my employer. _____
10. It will help me with other subjects. _____
11. I am interested in this course for other reasons not listed. I am listing these reasons here:

____

____

____

____

____

____

Appendix E: Student Attitudes towards CBJava Survey

Student Attitudes towards CBJava Survey Close Ended Questions

Answer each of the following questions by selecting one of the options provided writing it in the space provided.

A) if you strongly agree
B) if you agree
C) if you have no opinion
D) if you disagree
E) if you strongly disagree

1. _____ CBJava is an effective tool in learning object design. (note. For the students that are using the tool for inheritance the words "inheritance" will instead be "object design")
2. _____ CBJava helped my understanding of how real world problems can be conceptualized in Java classes.
3. _____ I had difficulty relating the examples to the instructional content within CBJava.
4. _____ I found that the ability to create concrete examples with expert feedback supported my understanding of the instructional content.
5. _____ CBJava supported my understanding of chapter topics by linking the examples to the related topics.
6. _____ I found the CBJava tool difficult to use.

Student Attitudes towards CBJava Survey Open Ended Questions

1. What did you like about the CBJava tool?
2. What did you dislike about the CBJava tool?
3. Did CBJava help you in your understanding of the subject matter? If so, tell us how it helped. If not, tell us how it hindered your understanding.

4. How do you think CBJava could be improved in the future?
5. Might a tool such as CBJava have helped your understanding in CM111 Introduction to Structured Programming? If so, tell us how it might help. If not, tell us how it would hinder your understanding.

Appendix F: Motivation Survey Open Ended Questions Feedback

Question #11: I am interested in this course for other reasons not listed. I am listing these reasons here:

Response(s)

- It's also a requirement.
- To expand my knowledge base. To expand my competence of all aspects of computing. To continue my understanding of program design.
- Major requirement.
- I am a chemistry and physics major interested in computational research.
- Required by major and will be needed and useful in grad school.
- Can apply my own practical uses to programming.
- It's part of the coursework.
- I love computers.
- Computers are the future and present of the global society. It is both exciting and nerve racking to be in this growing field.

Appendix G: Attitudes towards CBJava Survey Open Ended Questions Feedback

Question #1: What did you like about the CBJava tool?

Response(s):

- Ease of use. Examples and repaired examples.
- I thought the concepts were well explained.
- I liked how the examples were corrected and resubmitted. This allowed us to see errors and how they were fixed.
- Doing the examples helped a lot in understanding what was happening.
- Great concept.
- The posted examples could be good reference. The site was easy to use. It just wasn't useful.
- Not much, I prefer the lecture style.
- It encourages discourse between students by allowing us to submit examples.
- I like the whole online teaching concept. Wish all classes would move online.
- Posted examples from other students made it easier to understand through variety.
- The CBJava tool was pretty straight forward and also gave good examples.
- Helped with explaining inheritance.
- Good for seeing complete examples.
- It's nice to have things on screen categorized in more formal methods.
- I like the examples and the study tools on the site.

Question #2: What did you dislike about the CBJava tool?

Response(s)

- Having to be on the internet - just because I have a slow connection
- It was sometimes hard to read. Maybe a new more reader friendly graphical interface. But the content was just fine.
- Nothing, I still used the book and combined with CBJava I think I learned the subject well.
- Page is rather bleek and material could be reorganized.
- It was a waste of time and did no use it as a study tool. Doing assignments, reading the text and going to class (when possible) are efficient ways to learn the material. CBJava was just busy work.
- It feels more like an aid, rather than a full class.
- The only examples provided were exactly like the instructor's with the exception of one. Too difficult to find and work with. It should be integrated into WebCT somehow.
- It would be nice to bring up subj matter and print off the entire contents at one time (for latter referral), rather than each ind sect.
- Too narrow, many topics were repeated in different ways making it difficult to go to one place to find a non-specific answer.
- it was another study tool that repeated everything the book and presentation showed.
- Its role in the class seems undefined and it is hard to use.
- Navigation of site was awkward. A keyword search/topic search would help.

Question #3: Did CBJava help you in your understanding of the subject matter? If so, tell us how it helped. If not tell us how it hindered your understanding.

Response(s)

- It was rather neutral. Having the additional examples were nice, as was the extra credit.
- Sure it helped but I don't know if I can be any more specific. Maybe it's harder on a web page to skip ahead before you've finished reading the page you're on.
- Yes. It allowed me to further expand my knowledge on the subject as well as recognize mistakes (as stated in question one).
- The biggest factor for me was creating the example. It wasn't much but Helps provide a foothold of the subject matter.
- It helped, the feedback is great.
- It didn't help or hinder my understanding of the material. It might have been more useful to give more programming assignments.
- No, but it didn't hinder. I could find most of what it did in my book.
- Seeing examples by others.
- It helped through the other student examples but it could have expanded on the subject more. Reusing one example tends to get confusing and redundant.
- It helped a little. After viewing the presentation and reading the chapter, I just briefly skimmed over CBJava because the information was the same.
- Yes, gets to the point quicker.
- Glad to see complete programs as examples which is much more helpful than any book.
- While the examples on the site were informative I couldn't get a really good grasp of it until I had some face 2 face discussion of my doubts and misconceptions.
- Yes. It helped to organize how inheritance works and it helps to see example programs.

Question #4: How do you think CBJava could be improved in the future?

Response(s)

- Have smaller examples and more feedback. Have more examples in the text. Have a link that lets me download example with test class so I can see how it works.
- Other than maybe the interface, I liked the content how it was.
- Different example sections, chapters have a lot of small sections that could be exampled to learn.
- Make it a little fancier and organize material better. Get more examples. Highlight changes and repairs to clarify.
- The site could be used to discuss actual assignments rather than posting examples. Assignments are the key to learning.
- I think the concept is flawed. It works out to be little different than the text book.
- Integrate it into WebCT so everything is in one spot.

- More variety in teaching examples.
- It could be improved by making it the focus point of studying instead of the book or presentations.
- Don't know.
- User interface needs to be improved.
- Keyword search/Topic search. Chat/post boards for discussion of related topic

Question #5: Might a tool such as CBJava have helped your understanding in CM111 Introduction to Structured Programming? If so, tell us how it might help. If not, tell us how it would hinder your understanding?

Response(s)

- I would have received very little assistance from such a tool, but I am certain it could have helped a number of classmates.
- I think it would help in learning a new language. CM111 content is pretty simple - mainly learning about programming structures. I'm not sure how many examples of a while loop a person really needs to see. I certainly don't think it would hinder understanding. For me CM111 was about "getting my feet wet" and actually writing code.
- Yes the CBJava allowed me to see many examples and forced me to make an example. Seeing other people's ideas and mistakes really improved my knowledge on this subject.
- Much better examples on CBJava then can be provided in textbook.
- Yes, they should use it too.
- No, it would not have helped. A better text and more thorough programming assignments would have been of greater utility in CM111.
- I think CM111 should not use a similar tool. Programming is as much about mindset as it is about syntax and someone's first exposure to programming needs to be done in a way as to help with learning the mindset as well. That can only be done with real-life instruction.
- CM111 was way easy so I don't think students would use it.
- It would help with coding problems but as a replacement for an instructor it is too narrow.
- It might help because it is easy to show and demonstrate examples and how they work.
- Possibly - more interaction.
- I don't think it's appropriate for that class [cm111]. People taking this class have a better understanding of programming concepts. I think "newbie" programmers in cm111 would feel overwhelmed if given an example with no one to walk them through it.
- Yes, examples and the information is sound however post boards and threads would really have assisted in the Q & A.
- No. I took CM111 years ago and programming has changed a lot. I can't compare it to the type of programs we deal with in Java.