2008

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

DEVELOPMENT AND EVALUATION OF AN ADAPTIVE GRADING/LEARNING SYSTEM (AGLS)

Kevin Matthews

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

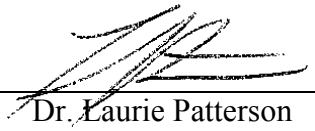
Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2008

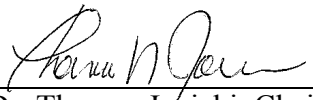
Approved By

Advisory Committee



Dr. Laurie Patterson

Dr. Ling He



Dr. Thomas Janicki, Chair

Accepted By

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

This research investigates the development and evaluation of an adaptive learning system based on pedagogy. Following implications found while surveying learning theory, the goal of this research is to improve the quantity, quality, and speed of feedback as it pertains to the grading of basic computer competency skills. Feedback has been identified as a key component of student success. This research builds upon the previous knowledge of the cognitive, behavioral, and resource-based views of learning. A system with a certain level of automation was developed that allows instructors to quickly grade multiple complex computer literacy assignments. This system was named the Adaptive Grading/Learning System (AGLS). Key to the success of the system was the ability to "learn" the correct and incorrect responses and store them for future use. To understand the impact of the AGLS on feedback, four hypotheses were created and experiments were developed to test them. The AGLS was shown to positively affect the quality and speed of feedback.

ACKNOWLEDGMENTS

The development and evaluation of the Adaptive Grading/Learning System (AGLS) was supported by a number of individuals whose participation was crucial in successful completion of the research and development.

Faculty members providing guidance and support included the committee members chosen for this research. Namely, Dr. Thomas Janicki chaired the committee and offered assistance in choosing a topic of research, providing additional funding for research, and participating in the entire process of development. In addition, committee members included Dr. Ling He and Dr. Laurie Patterson.

There were also several other faculty members who adopted the AGLS for use within their courses. These professors included members of the committee. Those who were not members of the committee included Dr. Judith Gebauer, Mr. Ed Topor, and Dr. Ulku Yaylacicegi. The grading of assignments was necessary for data collection needed for analysis for the research. Graduate teaching assistants from the Department of Information Systems and Operation Management also aided in the grading and passive data collection. Matthew Mascherin, Sarah Peck, and Ashley Vereyken were the graduate assistants who had the most exposure within the system.

Finally, faculty members who did not use the system but participated in the expert panel study used for analysis are to be thanked. This data allowed another hypothesis to be added to the research conducted. Dr. Douglass Kline, Dr. Bryan Reinicke, and Dr. Albert Ritzhaupt were the three members of the expert panel.

Without the help of those mentioned above, the successful development and evaluation of the AGLS would not have been possible.

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

This research investigates the development and evaluation of an adaptive learning system based on pedagogy. Following implications found while surveying learning theory, the goal of this research is to improve the quantity, quality, and speed of feedback as it pertains to the grading of basic computer competency skills.

Current learning theory provides insight into what affects the success of students in introductory computer courses. This theory proposes that the number of challenging assignments must be increased and timely feedback must be received in order to increase knowledge and retention. In the current situation of increased enrollments per class and a static number of instructors, these goals cannot be achieved. This article details the supporting learning theory and an analysis of the issues currently facing instructors in the academic setting.

A system with a certain level of automation was developed that allows instructors to quickly grade multiple complex assignments in Microsoft Access 2007 and Excel 2007. The system has been named the Adaptive Grading/Learning System (AGLS). The AGLS allows a way to provide the grading and feedback that instructors need without restricting the type of problems and cases that can be used. The AGLS also allows instructors to input an assignment in which the system will "learn" the correct and incorrect answers and provide feedback accordingly. The process of development and evaluation are detailed in this paper.

In addition to the development and implementation, four hypotheses were developed and evaluated using data collected by the AGLS. These four hypotheses pertain to the quantity of feedback, quality of feedback, response time of grading, and amount of grading errors produced by the system. Data analysis and conclusions are given with each hypothesis discussion. The hypotheses developed and analyzed are directly related to learning theory.

II. LEARNING THEORY CONCEPTS

This section builds a foundation of specific learning theory concepts that analyze how a student learns and what the academic community can do to increase the likelihood that learning will occur. Based on these learning theory concepts, several null hypotheses were developed that are linked to the implementation of the systematic grading system called the Adaptive Grading/Learning System (AGLS).

A student's success is influenced by the ability of the instructor to present new information and evaluate the student's understanding of the information. This process requires the student to learn the material covered by the instructor. Robert Gagne (1965, 1985, 1988; Gagne et. al, 1992) defines a list of nine elements that should be present in any lesson in order for learning to occur. These nine elements form the framework for cognitive learning theory. They are:

- Gaining attention ("reception")
- Informing learners of the objective ("expectancy")
- Stimulating recall of prior learning ("retrieval")
- Presenting the stimulus ("selective perception")
- Providing learning guidance ("semantic encoding")
- Eliciting performance ("responding")
- Providing feedback ("reinforcement")
- Assessing performance ("retrieval")
- Enhancing retention and transfer (generalization")

Of the nine "conditions for learning" that Gagne et al. (1992) provide, other research shows that eliciting performance and practice from the student ("responding") and providing adequate feedback ("reinforcement") are the events most directly connected to student

success (Martin et. al., 2007).

"Responding" is required from learners after they have been given sufficient material to comprehend an objective (Gagne, 1985). Elicited performance on the student's part is required when practice is included in a lesson. This form of practice implies an active response to the material provided. For example, in a database lesson, "responding" might require a student to create a query that will count the number of records in a table in order to demonstrate his/her comprehension of this newly introduced concept. None of the other nine elements of instruction mentioned by Gagne require the student to actively "respond" to the materials presented by instructors (Martin et. al., 2007). The presence of this element of instruction ("responding") enables the student to reinforce his/her understanding. Effective practice should parallel the assessments that are used to test skills and the knowledge reflected in an objective (Reiser & Dick, 1996). Gagne continues to show that once practice has been completed, the student should have been adequately challenged to apply the new knowledge in a real life situation. Another positive result of learning through practice is the motivation achieved through the active participation and increased confidence in the objective tested (Dewald, 1999).

Another learning theory, known as behavioral learning theory, includes several characteristics that should be present in an effective instructional design. These principles are contiguity, repetition, and feedback (Gagne et. al., 1992). Contiguity is achieved when the response elicited from a student follows the presentation of material as closely as possible. Students should be expected to perform a "responding" activity (elicited performance) immediately after a learning objective is covered. Repetition increases the likelihood that a student will retain information presented during a lesson. This can be achieved by an increase in the number of assignments that allow a student to respond to many similar questions or

tasks. Feedback occurs when the assignment is analyzed and answers are identified as correct or incorrect. Not only is it important to identify answers as correct, but the explanation of an incorrect answer and the supporting rationale are essential. Providing and explaining the correct answer and explaining the faults of an incorrect answer are helpful when learners answer incorrectly (Kulhavy, 1977). Phillips et al. (1988) note that adequate feedback decreases the repetition of incorrect answers in the future and increases the probability of repeating correct responses.

Likewise Rakes (1996) recommends increasing a student's success through the addition of practice and feedback through a shift from the traditional theories of learning (cognitive and behavioral) to a resource-based view of learning. The resource view of learning involves the role of an instructor changing from one of an expert dispensing knowledge to one of a guide providing resources. As more online or web-enhanced courses become available, the need for this theory of instruction increases. Table 1 provides a further comparison of the traditional view and resource view of learning.

The resource-based learning approach increases the likelihood that a student will grasp the concepts presented because the responsibility of discovery is given to the student. The student must take ownership of the material presented and use the resources provided by the instructor in order to complete a lesson. Similar to cognitive and behavior theories, to be effective the resource view of learning requires an increase in the number of problems, assignments, and exercises (Rakes, 1996). This need is discussed later, including the problems of corresponding solutions that arise from these requirements.

Other researchers concur with Gagne and Rakes. Practice ("responding") and feedback ("reinforcement") have been found to be the most-influential factors in Gagne's nine elements of instruction. Martin et. al. (2007) performed a study in which six different

versions of an instructional program were developed and presented to a variety of students enrolled in computer literacy courses. Five of these six program versions neglected to include one of Gagne's events of instruction. Four of the versions included practice and feedback to the students. Students were given a pretest of the material, presented with the lesson using one of the six instructional programs, and then given a posttest to mirror the pretest. The results of the study showed that student success (measured by performance on the posttest) was significantly higher in students who were given the practice and feedback sections of the program. Student performance suffered when the administered program neglected to provide practice and corresponding feedback.

Table 1: Traditional Versus Resource-Based Learning (Rakes, 1996)

Traditional Learning	Resource-based Learning
Teacher as an expert model	Teacher as a facilitator/guide
Textbook as primary source	Variety of sources/media
Facts as primary	Questions as primary
Information is packaged	Information is discovered
Emphasis on product	Emphasis on process
Assessment is quantitative	Assessment is qualitative/quantitative

Student attitudes were also measured in the Martin et. al. study. As was the case with the success rates, attitudes of students who experienced practice and received feedback were more positive than those who were not provided practice and feedback. This thesis builds on previous research that found that the probability of retention of a correct response is increased and the probability of incorrect responses is reduced when given proper practice

(Phillips et. al., 1988).

In summary, a study of learning theory shows that a student's success is enhanced when he/she is given challenging, real-world practice assignments with rapid feedback. The following key concepts were used in the development of the AGLS:

- To be effective, the student's response to concepts should immediately follow instruction.
- Multiple assignments of similar nature should be presented repetitively to reinforce new material presented during a lesson.
- Quick and customized feedback allows a student to identify correct answers and see errors in incorrect answers.
- The best way to reach this goal of student success is to adopt a resource-based approach to learning.

III. NEED FOR AN ADAPTIVE GRADING/LEARNING SYSTEM (AGLS)

The requirement of mastering personal productivity applications – defined as word processing, spreadsheet, presentation, and database software – is growing for not only students, but all individuals who wish to compete in an increasingly technology-driven world economy. When defining the ten concepts that students need in order to gain fluency in Information Technology, the National Research Council (NRC) and National Science Foundation (NSF) indentified four that implicitly relate to the need for fluency in personal productivity products (National Research Council, 1999). In addition to this call for fluency, many colleges and universities require computer literacy for their degree-seeking students. The UNC Wilmington 2007-2008 undergraduate catalogue states

> *"the University requires that all students prior to graduation develop competency in basic computer skills including … facility with standard applications, and awareness of legal and ethical issues. Students in each major must satisfy the requirements in computer competency as specified by that major" (University of North Carolina Wilmington Undergraduate Catalogue, 2007, p.117)*

These standard applications are found in the typical "office suite" of software known as personal productivity products.

The results of these requirements imposed for computer competency have impacted not only students, but instructors as well. Enrollment in introductory level courses that cover personal productivity software has increased because of the requirements found at many institutions. This increased enrollment has not been matched by a corresponding increase in faculty. In fact, instructors have had to utilize new classroom environments to cope with increased enrollment. In the Spring 2008 semester, UNCW's Department of Computer

Science incorporated five online Introduction to Computers & Applications (CSC 105) sections. The Department of Computer Science also offered three sections of this course in a combined auditorium setting with one instructor. Similarly, UNCW's Department of Information Systems and Operations Management offered one online section and three sections of the Introduction to Information Systems and Technology (MIS 213) course in an auditorium setting with one instructor. In total, there were sixteen CSC 105 sections with approximately 350 students enrolled and fifteen sections of MIS 213 with approximately 450 students enrolled. The types of assignments for these introductory courses require a substantial amount of time to grade. With the increase in enrollment, at least a linear increase in the grading time required is expected and time becomes even more limited for many instructors (Kay, 1998).

The burden placed on instructors has resulted in assessment and grading processes that do not support the previously discussed learning theory. Because of the amount of time required to grade assignments, many instructors have chosen to give more mundane assignments that do not adequately challenge the students. Syllabi from UNCW's CSC 105 and MIS 213 courses show that many instructors have also opted to require only a few hands-on assignments. This is most likely done in order to reduce the overall time spent grading.

The feedback given to students – measured by an average character count as well as time for feedback to be returned – has also suffered and may not even provide the needed input for reinforcement of student success. With the offering of courses online or in an auditorium setting on the rise, the need for customized feedback is increasing. In a sampling of data taken from course management software used by Fall 2007 introductory classes, 130 of the 429 graded assignments (30.3%) had no related comments from the instructor. This sampling excluded assignments that received a perfect score. Instructors are restricted by the

amount of assignments given because of the time needed to grade. The result is often delayed feedback. Data from the same sample reveals that the average difference of time between an assignment's due date and date graded was 28 days. It is not uncommon for a student to receive a graded assignment more than four weeks after the assignment was submitted.

It may also be the case that graduate assistants have graded assignments in conjunction with the instructors. This fact can lead to inconsistent grading and feedback. Each graduate assistant may have a different standard and weighting for an assignment. For example, one grader may deduct three points for an incorrect formula in an Excel spreadsheet and provide no feedback while another may deduct only two points and comment that the student should have used an absolute reference instead of a relative reference. Whether there is one grader per assignment or several graders working together, there is always an increasing chance of human error. Kay (1998) states "to preserve consistency…it is best for a single individual to grade every student's response to a given question." This is simply not possible with the increased enrollment faced by instructors for introductory computer courses. Grading errors can often go unnoticed and may lead a student to believe his/her answer to be correct when it is actually incorrect. As shown in the learning theory review, identification and correction of errors holds a direct correlation to an increase in a student's learning.

Finally, because of the decrease in time the instructor has available, the lack of adequate feedback, and the failure of the current processes to allow practice and promote success, a student's attitude and motivation may suffer greatly (Martin et. al., 2007). This decrease in interest often results in an increase in the occurrences of plagiarism that can often go unnoticed by assignment graders (Miller et. al., 1981). The problems that arise from the current situation of computer literacy courses are summarized in Table 2.

Table 2: Summary of Current Problems in Introductory Computer Courses

Current Problems Identified in Literature Review and Current Practices
Few assignments per learning objective
Less challenging assignments
Slow feedback
Generic (or no) feedback
Many different graders
Inconsistent grading
Inconsistent feedback
Grading errors
Increased plagiarism

The issues found in Table 2 do not promote learning. These current problems are in fact contradictory to the requirements for learning found in learning theory today. In order to reach student success, the resource-based learning approach states that there must be an introduction of more exercises that present a real-world challenge to the students (Rakes, 1996). If instructors were to adopt this ideal learning style and introduce more exercises in their current state, even more issues would arise. The time required for grading assignments would become more overwhelming than that of the current status. Not only must the number of assignments be increased, but there must also be sufficient, timely feedback in order to achieve the best potential for learning. Due to time constraints, this task is almost unachievable now and would only prove to be harder to accomplish if the required increase in the number of exercises occurred as course enrollment continues to increase.

Based on these issues, a need for a system exists that can relieve the pressure caused by the coupling of stressors such as increasing enrollment and time-intensive grading. The aforementioned problems call for a system with a certain level of automation that will allow instructors to quickly grade multiple complex assignments with quality feedback. Instructors need access to a larger set of assignments that can be graded quickly and with increased accuracy. If these needs can be met, then students can be presented with increased practice and rapid feedback required to promote learning.

IV. SYSTEM BENEFITS TO STAKEHOLDERS

In order to meet the need of students and instructors, a system was developed known as the Adaptive Grading/Learning System (AGLS). This system consist of modules that provide automated grading of Microsoft Excel 2007 and Access 2007 assignments with personalized and rapid feedback, shared assignment libraries, and plagiarism detection.

The AGLS was developed to serve as a solution to the problems identified in Table 2. The benefits from the use of the AGLS extend to the instructors, the students, as well as other faculty and staff members. More of the resource-based learning style may be implemented through the adoption of this AGLS as the means of incorporating practice and grading for computer literacy courses. It was anticipated that the overall result of this adoption would be increased student fluency. This is achieved through the possibility of increasing the number of exercises of a more challenging nature and the usage of personalized and timely feedback that can be provided.

The AGLS may counter balance the problems caused by an increase in enrollment and a static number of instructors. The automated processes of grading reduce the time-intensive nature of grading currently faced by instructors. As the burden of grading is reduced, instructors are able to assign more exercises and are given the ability to choose exercises previously used by other instructors that are gathered in an assignment library. The AGLS also allows the complexity of exercises to be increased. This increase in complexity serves to challenge students and increase the likelihood of their success and learning.

Solutions to other potential learning impediments may result through the implementation of the AGLS. Feedback is provided for each item within an exercise and can be personalized for each answer given. The system compiles a list of correct and incorrect responses and allows the instructor to identify feedback accordingly. The potential for

enhanced and personalized feedback provided by the AGLS is ideal for student achievement. Another benefit to students is the resulting timely feedback of assignments. The automated portions of the AGLS scan assignments, grade them appropriately, and provide the corresponding, instructor-supplied feedback. These instructor-assisted portions of the system prove to be quicker than any hand-graded efforts. The AGLS has an adaptive set of correct and incorrect answers for each assignment that is refined every time the assignment is graded.

On top of the benefits already discussed, the AGLS contains components that allow the tool to be scaled across multiple disciplines. The AGLS is generic and was not developed to be specific to any one domain, discipline, or set of assignments. The Microsoft Office 2007 suite of software has been chosen as the standard at UNCW and is used in the many diverse disciplines. The AGLS allows instructors access to its different modules through the Internet. Instructors and students of courses in all science, humanities, and business curriculums could benefit from the incorporation of the AGLS into their courses. Many portions of the system allow benefits that can be easily and effectively scaled.

In summary, the AGLS allows a non-discipline specific way to provide rapid and customized feedback for practice assignments. This allows instructors to assign more assignments and offer timely feedback to students. The AGLS is a much needed step towards the desired resource-based learning approach that current learning theory suggests for student success.

V. SYSTEM DESCRIPTION

The Adaptive Grading/Learning System (AGLS) is comprised of distinct grading, reporting, and library components. These components were developed as a series of modules available to faculty, staff, and students. The component-based nature of the design allowed each part of the system to be constructed independent of the other parts rather than as a single complex system. This also allows other components to be developed and added to the system after this primary implementation. Initially, the components were developed to provide services to instructors grading Access 2007 and Excel 2007 assignments. The major system attributes are the adaptive grading process, the input process, the assignment library, and plagiarism detection.

A. Adaptive Grading Process

Since the process for grading documents created in Microsoft Office applications is similar, the AGLS consists of a universal process for requesting grading services. Each assignment requires a grading key that can be used as the "best response" to the proposed problem set. This input identifies what needs to be graded (e.g. what cell and formula should be checked in an Excel file or which field should be the primary key in an Access table). The student file to be graded against this key is also required. And lastly, it must be known which grading service will be used (proposed Excel or Access). The grading component of the AGLS takes these inputs and produces a raw score for each input file, a set of comments to be used as feedback to the student, and a set of correct and incorrect answers used. Figure 1 shows the input and output of the adaptive grading process of the AGLS.

The process used for grading is results-based. This is a key difference between the AGLS and other available systems (see Section VI). While the actual answer that the student has reached is graded, the process or mouse clicks required to achieve an answer are not.

Each gradable item (identified by the instructor) is associated with two lists. A list of correct responses allows acceptable answers to be marked accordingly. For example, in an Excel spreadsheet, a student may multiply cells C5, C6, and C7 together by using =C5 * C6 * C7 or using =C7 * C6 * C5; both are correct. A list of incorrect responses is also coupled with appropriate feedback for the student.

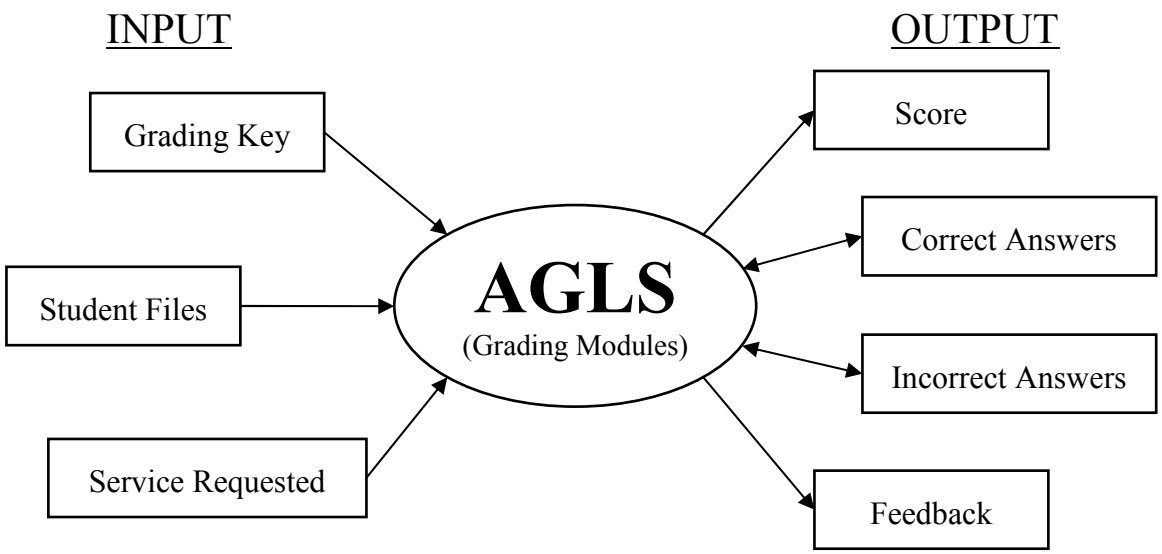


Figure 1: Input/Output Diagram for Adaptive Grading Process of the AGLS

Appendix A diagrams the process flow of all adaptive grading functions of the AGLS. The process of the grading component of the AGLS is as follows. First a comparison is made between a student's answer and the list of correct responses. If no match is found, the item is compared to the list of incorrect responses. If the answer has not been previously flagged as correct or incorrect, the instructor is prompted to identify the student's response as either correct or incorrect (Figure 2). The answer is added to the appropriate list. In the same prompt, the instructor may signify if he/she wishes to give partial or extra credit for the response and how much credit to award. Extra credit is represented by a number of points to

add to a task's weight while partial credit is a percentage of the task's weight to award for the given answer. With every answer the instructor is also prompted to provide feedback that can be associated with an incorrect answer. When setting up a task, the instructor provides default feedback. This default is filled in for the instructor when grading and can be changed if desired.

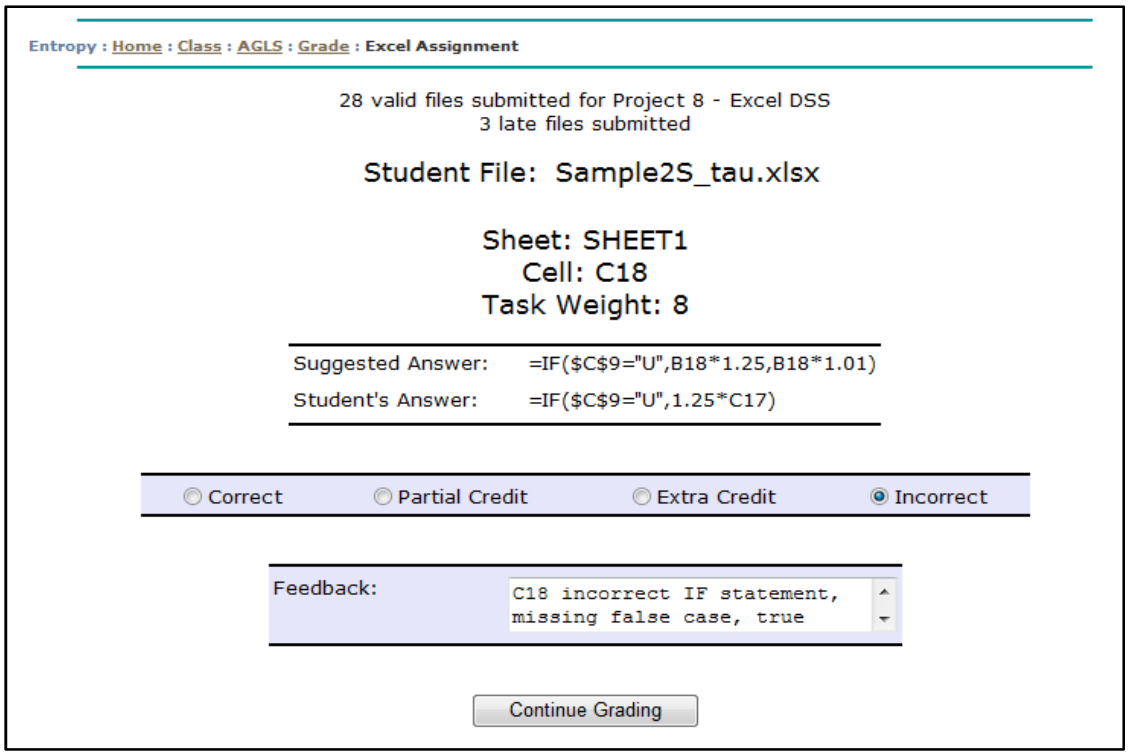


Figure 2: Instructor Prompt to Identify Student Answer as Correct or Incorrect and Associated Partial/Extra Credit and Feedback.

Once an answer has been flagged as correct or incorrect and added to the appropriate list, any future response of the same answer (given by other students) will not need to be reviewed. To prevent errors, instructors are given the option to review the answers previously flagged as correct or incorrect. If a mistake is found, the instructor will be able to reevaluate an answer as needed. Since the grading key can be changed at any time, files can be graded multiple times. The grading key for each assignment will become more and more refined and

the manual inspection required by an instructor will decrease over time.

B. Input Process

The AGLS's web accessibility provides an easily accessible means of grading for instructors. The system is comprised of a series of dynamic web pages that allow the instructor to create a new library assignment, create a grading key, and initiate the grading process. Each library assignment can be comprised of multiple tasks to be graded. A task may be, for example, to enter a formula into a spreadsheet cell or to set a data type of a database table field. The dynamic nature allows the instructor to identify each task to be graded. With each task identified, key answers are given for all required items. Once all tasks and items have been created, the library assignment is ready to be matched with one or more course assignments and auto grading can commence.

C. Assignment Library

In addition to this means of building library assignments, instructors have access to assignment library modules in which previously created library assignments can be stored and shared with other instructors. Each library assignment includes the grading key and the system instructions for grading (correct and incorrect answer lists with associated feedback). The availability of the assignment library allows a larger problem set for instructors with the ability of rotation of assignments between semesters. This problem set incorporates the correct and incorrect answers provided collectively as an assignment is used in several classes. The AGLS's process for grading allows the grading template given for an assignment in the library to "learn" over time with a longer list of flagged correct and incorrect answers and feedback.

D. Plagiarism Detection

In addition to the grading services, instructors will have access to the plagiarism

detection tools provided by the AGLS for particular Office products. The methods of plagiarism detection differ between Access and Excel.

For Access, instructors are required to flag a table or query of interest for plagiarism detection. The internal creation times of these tables and queries are stored in the database for later comparison. Once the files have been graded, the instructor can view all creation times for the tables and queries of interest. Those objects that have the same creation time are flagged by the plagiarism detection mechanism to be investigated. From here, the instructor can evaluate each student's file and determine what actions need to be taken.

For Excel, students are required to download templates provided by the instructor. These are embedded with the student's id in a hidden, password-protected sheet. When an assignment is uploaded, the AGLS compares the embedded downloader's id with the id of the student who is uploading the assignment. The file is flagged as a potentially plagiarized file if there is a mismatch from the two ids and the student's feedback includes a request to meet with the instructor. In this case, both individuals involved in the plagiarism are identified to the instructor.

VI. ALTERNATE SYSTEMS

There are alternative grading systems currently available to instructors, but each has its own limitations. These systems can be grouped into categories in which shared pros and cons exist. The categories include case-based auto graders, procedural based grading systems, and simple test-bank systems. A discussion of each category with system examples follows. A comparison chart of benefits in each alternative category and the AGLS can be found in Table 3. Drawbacks for comparison are provided in Table 4.

A. Case-based Auto Graders

An example of a case-based auto grader is CASEGRADER by Thomson Course Technology. Instructors are provided with a set of cases that can be instantly graded by the CASEGRADER system. This type of alternative system offers challenging, multi-step, realistic problems that students may submit to be automatically graded. Feedback is instantaneous and based on incorrect responses. Students are informed of their grade and feedback immediately following their submission of an assignment. For example, the CASEGRADER system allows students to submit their completed cases to an online portal known as CoursePort. Feedback is immediately rendered and inserted directly into the student's file. Limitations to this system include the inability of instructors to create their own cases (Crews, 2008). CASEGRADER offers twelve (12) cases for the Office 2007 release. If all sections of a course use the same limited cases, an increase in student plagiarism could occur. Currently, CASEGRADER is limited to Microsoft Excel.

B. Procedural-based Grading Systems

Procedural-based grading software includes systems such as SAM 2007 by Thomson Course Technology or SNAP by EMC Paradigm Publishing. These alternative systems are applications that grade student responses based on the procedure used to reach the answer.

The application may either be a web system or a software application that simulates the environment of Microsoft Office programs in order to provide a hands-on experience for the students. These systems usually incorporate small, "toy" problems that attempt to reinforce a procedure to be remembered. For example, these graders may ask students to bold or underline a cell or enter a simple SUM function. Few complex problems exist in the database of questions for these graders. Many procedural-based graders include test-bank systems that can be used to create pretests, posttest, or quizzes used to evaluate a student's learning.

C. Simple Test-Bank Systems

Simple test-bank systems normally exist within other systems such as Blackboard or SAM 2007. These systems provide the instructor with the ability to create multiple choice, fill-in-the-blank, or paragraph/open-ended questions. Questions can be developed and stored for later use. Automated grading is provided by these alternative systems for multiple choice and fill-in-the blank responses, but paragraph questions require manual grading by an instructor or grader.

D. Comparison of Benefits and Drawbacks

As seen in Table 3, the AGLS is the only system that provides all additional features in order to ease the burden on instructors and still promote student learning. The AGLS's grading modules can provide the grading of multi-step, comprehensive problems or simple, one-skill problems. The flexibility of the assignments that instructors can define is a unique system advantage. Also, the AGLS is unique in its ability to provide an assignment library that can be shared and expanded across several domains and courses. With the capability of "learning" new answers and expanding the provided answer bank, the AGLS stands above the alternatives presently on the market. Not only do the advantages of the AGLS stand strong, but the disadvantages and limitations depicted in Table 4 are not present. These

drawbacks do however exist throughout other alternative systems.

Table 3: Comparison of AGLS and Alternative System Benefits

Benefits/Features	AGLS	Case-based	Procedural	Test-Bank
Challenging, real-world problems	■	■		
Automated grading	■	■	■	■
Consistent grading	■	■	■	■
Instant feedback	■	■	■	■
Customized feedback	■			
Web interface/portal	■	■	■	■
Multiple skills assessed concurrently	■	■		
Hands-on experience	■	■	■	
Smaller one-skill problems	■		■	■
Question/assignment library	■		■	■
Reduced preparation/paperwork time for instructor	■	■	■	■
Availability of student reporting	■	■	■	■
Expandable answer banks	■			
Repository for file submissions	■	■		
Plagiarism detection	■	■		
Instructor created exercises	■			■

Table 4: Comparison of AGLS and Alternative System Drawbacks

Limitations	AGLS	Case-based	Procedural	Test-Bank
Answers must be exact matches		■	■	■
Limited number of cases		■	■	
Textbook/supplemental required		■	■	
Software must be installed			■	
"Simulated" environment			■	
Other purchases required		■	■	■

VII. METHODOLOGY AND PLAN FOR DEVELOPMENT

This section will detail the software development methodology that was adopted in order to complete the software development. The software development life cycle includes the following phases: project planning, analysis, design, implementation, testing, and support. Before making the decision of which system development approach to use, each methodology was weighed according to its strengths and weaknesses. Three common methodologies were considered. These methodologies were: the waterfall approach, the spiral life cycle model, and the unified process. Strengths and weaknesses are discussed and then the decision used in the development of the AGLS is given.

A. The Waterfall Approach

The waterfall approach is a predictive methodology of software development. Predictive methodologies appeal to systems in which the domain is well known and requirements can be determines at the onset of the development process. The waterfall approach demands that each phase in software development be executed in direct succession and without backtracking. Once one phase has been completed, the specifications created are frozen and cannot be altered (Satzinger et. al., 2005). Figure 3 (Satzinger et. al 2005, p. 41) depicts the waterfall methodology.

B. The Spiral Life Cycle Model

The spiral life cycle approach to software development incorporates a more adaptive approach to development. Adaptive approaches assume the phases of software development cannot be independently executed and must be improved over time in conjunction with the other phases. An adaptive methodology is ideal for a development project in which the requirements cannot be known up front, analysis is a multi-step process, and the design may frequently change. The spiral life cycle allows multiple iterations that include all

development phases. In addition, the spiral approach also embraces the notion of prototypes. At the end of each iteration in this methodology, a prototype is developed that can be tested and analyzed in order to refine the requirements and specifications gathered in previous iterations (Satzinger et. al., 2005). Figure 4 (Satzinger et. al 2005, p. 43) depicts the spiral life cycle approach.

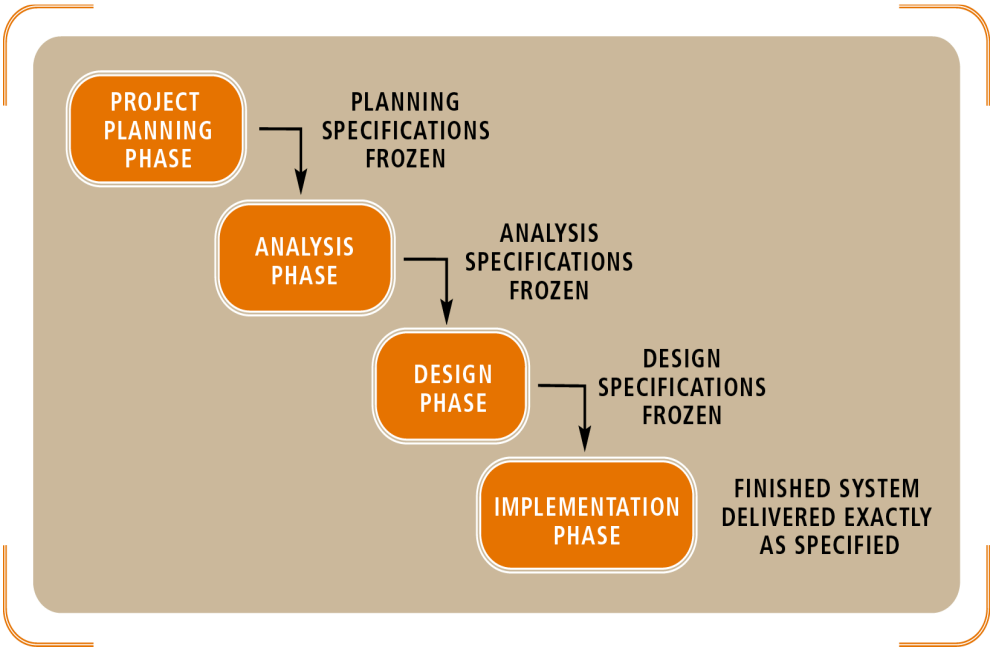


Figure 3: The Waterfall Software Development Methodology

C. The Unified Process

The Unified Process is an object-oriented development methodology that simplifies the process of planning and managing iterations of system development. This adaptive approach uses four phases that are use case driven. This means that Unified Modeling Language (UML) use cases provide the foundation for defining functional requirements and designs throughout the development life cycle. These phases are inception, elaboration, construction, and transition. To allow better time management and easier refinement, each phase can be further broken down into one or more iterations. Figure 5 (Satzinger et. al 2005)

shows the breakdown provided by the phases of the Unified Process.

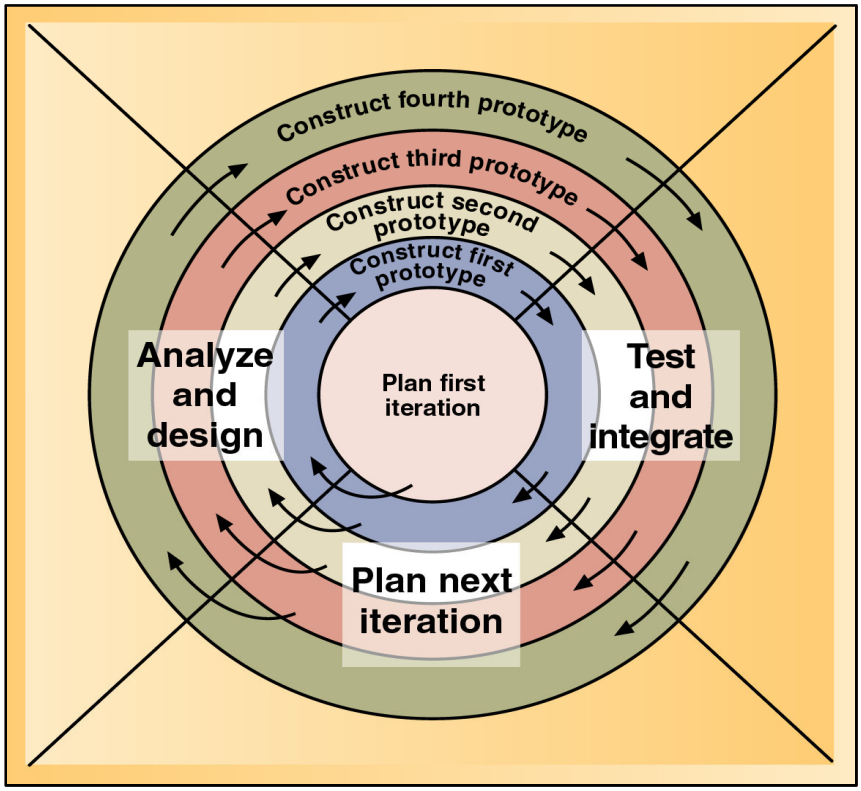


Figure 4: The Spiral Life Cycle Methodology

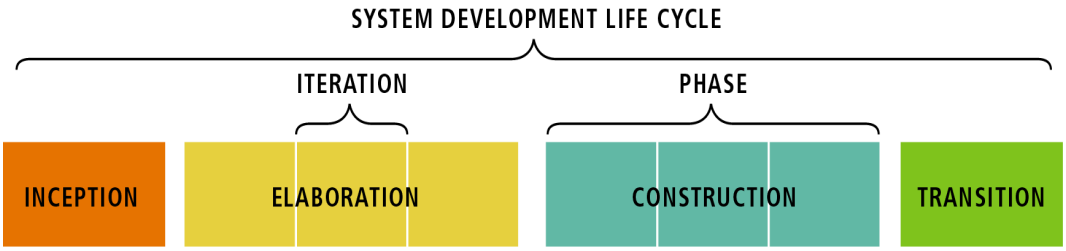


Figure 5: Phases and Iterations of the Unified Process

The Unified Process uses not only phases, but also several disciplines in order to determine focus and drive during the life cycle of development. The disciplines of the Unified Process include business modeling, requirements modeling, design, implementation, testing, deployment, project management, configuration and change management, and

environment. Each discipline is present throughout all phases of development, but a stronger focus is provided within the separate phases based on the goal of the phase. Figure 6 (Satzinger et. al 2005, p. 54) depicts a standard representation of the Unified Process used for a seven iteration project and the importance of each discipline to each iteration and phase. The presence of each discipline varies by iteration and phase (Satzinger et. al., 2005).

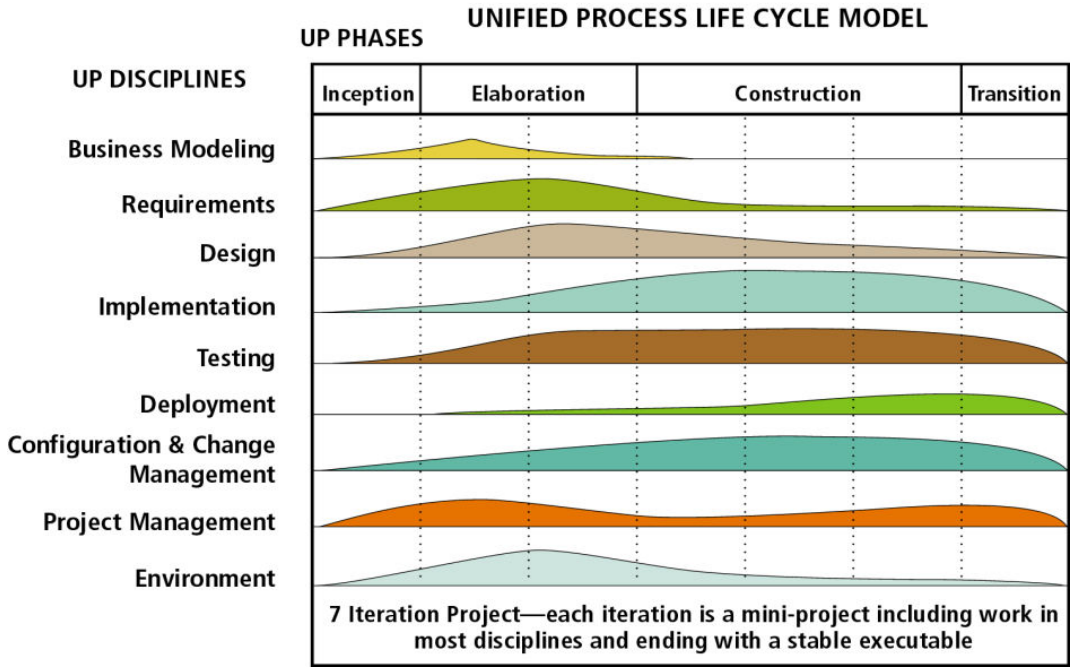


Figure 6: The Unified Process as Applied to an Iterative Project

D. The Methodology for the AGLS

The AGLS required an adaptive life cycle approach to system development. The specific requirements, design, and analysis could not be completed up front while still meeting the desired quality of modular web development. This initially ruled out the waterfall approach because of its predictive requirements. Also, because of the nature of the domain and the likelihood that changes in design, testing, and specifications could have occurred during the development of this system, a phased methodology was more applicable.

The spiral life cycle model has the characteristics that allow phased development, but the portions of each phase in the spiral methodology are independent. The best methodology to be used during development of the AGLS was the Unified Process. This methodology allowed the required iterative phases to occur, each including a portion of the defined Unified Process Disciplines. The extended existence of the aforementioned disciplines was ideal for the AGLS development. The focused and organized object-oriented nature of the Unified Process better served the stakeholders and developers during this project.

VIII. DEVELOPMENT AND IMPLEMENTATION

To satisfy the requirements of the development of the AGLS, the Unified Process was chosen as the software development methodology. This section details the development and implementation of the AGLS and how the steps taken adhere to the Unified Processes four phases: inception, elaboration, construction, and transition. Figure 5 depicts these four phases.

A. Inception Phase

The inception phase is used to set the framework for a system development project (Satzinger et. al, 2005). This phase involved identification of benefits and their associated costs, defining scope, identifying key requirements of the system, and determining the technology to be used. The inception phase consisted of one iteration, which is denoted as I1.

Iteration I1. I1 initially began with a literary review of learning theory concepts which is cataloged in Section II. This review revealed the need for students to be challenged by real-world practice assignments. Sources show that these assignments enhance a student's success. In identifying the benefits during the inception phase, faculty from the Department of Computer Science and the Department of Information Systems and Operations Management were interviewed. Assignments, syllabi, and textbooks that were given in introductory computer courses were reviewed. The need for a system with a certain level of automation that would allow instructors to quickly grade multiple complex assignments was found. The need and benefits that were identified are further explained in Section III and Section IV.

In addition to helping identify benefits, faculty interviews also aided in defining scope and determining the key requirements for the AGLS. Based on the needs defined, the biggest burden in introductory courses was found to come from grading Microsoft Access

2007 and Excel 2007 assignments. Initially, the project considered that all of the Microsoft Office 2007 suite programs were to be included, but given the focus of the introductory courses, scope was narrowed to only Access and Excel. Using assignments and faculty as a guide, a list of gradable tasks was developed for both Access (see Appendix B) and Excel (see Appendix C). For example, formulas and cell values were seen as a necessary requirement for grading an Excel file. The plan of action that was adopted included a modular approach to development. This would allow for new components (either gradable tasks or Office products) to be seamlessly integrated into the AGLS as they are developed. The key requirements that were developed can be found in Section V.

B. Elaboration Phase

The elaboration phase serves many purposes in a software development project (Satzinger et. al, 2005, p. 46-47). This phase involved finalization of scope and requirements as well as analysis, design, and implementation of the core architecture and major components of the system. The elaboration phase consisted of five iterations which are denoted as E1, E2, E3, E4, and E5.

Iteration E1. The first iteration of the elaboration phase (E1) involved identifying the data storage needs of the AGLS and consisted of mostly design and concept. In order to ease the transition burden placed on instructors and students, the AGLS was integrated into UNCW Cameron School of Business's in-house developed course management software, Entropy. Entropy provided the backbone for the AGLS that included course creation, Assignment creation and editing, template downloading, Assignment uploading, and grade storage. The modules for the AGLS remained separate, but integrated throughout the development process.

The grading process and storage needs of the AGLS were explored and the database –

tables and relationships – was designed. The database design for the AGLS can be found in Appendix D. Note that this diagram is the final revision of the AGLS database. The basic structure of data storage was constructed. An instructor can create an Assignment that will be associated with a Library Entry. The ability to share Assignments among a community of instructors was provided by the Library Entries. The availability of these modules will allow the AGLS to be extended across multiple courses and even shared between several disciplines. These libraries allow instructors to leverage the assignments already processed by the Intelligent Grading Modules. These modules allow instructors to see previously used Assignments. Each includes the refined grading key with related feedback. Library Entries are shared among all instructors using the AGLS until an association is made to an Assignment. At this point, the Library Entry is copied (either in whole or in part) and will remain apart from other Library Entries for individual instructors.

A Library Entry consists of the Tasks that will be graded by the AGLS. Each Task is divided into a series of Items to be graded. Items serve to direct the AGLS in the sequence of events that will determine if a Task has been completed. Answers are provided for each Item given. For example, a Task to be graded may be a field type within Access. This Task is divided into three Items: table name, field name, field type. The Key Answers for these Items could be “Person”, “FirstName”, and “Text” respectfully. Figure 7 shows the relationship of Assignments, Tasks, Items, and Answers.

The AGLS required as much extensibility as possible; therefore the database was designed in such a way that new Library Entries, Tasks, Items, and Answers can be easily added, removed, activated or inactivated. Once the database design was finalized, the initial screens to be used as the user interface were sketched and the flow of information from the user interface to the database was investigated. It was evident that object-oriented

programming would be adopted and objects were designed to mirror the information being stored in the database. The initial object classes included a Task object, an Item object, and an Answer object. This practice allowed for the easiest implementation for data flow without creating an overbearing burden on the database. The creation of the database design, user interface screens, and object class designs marked the end of E1.

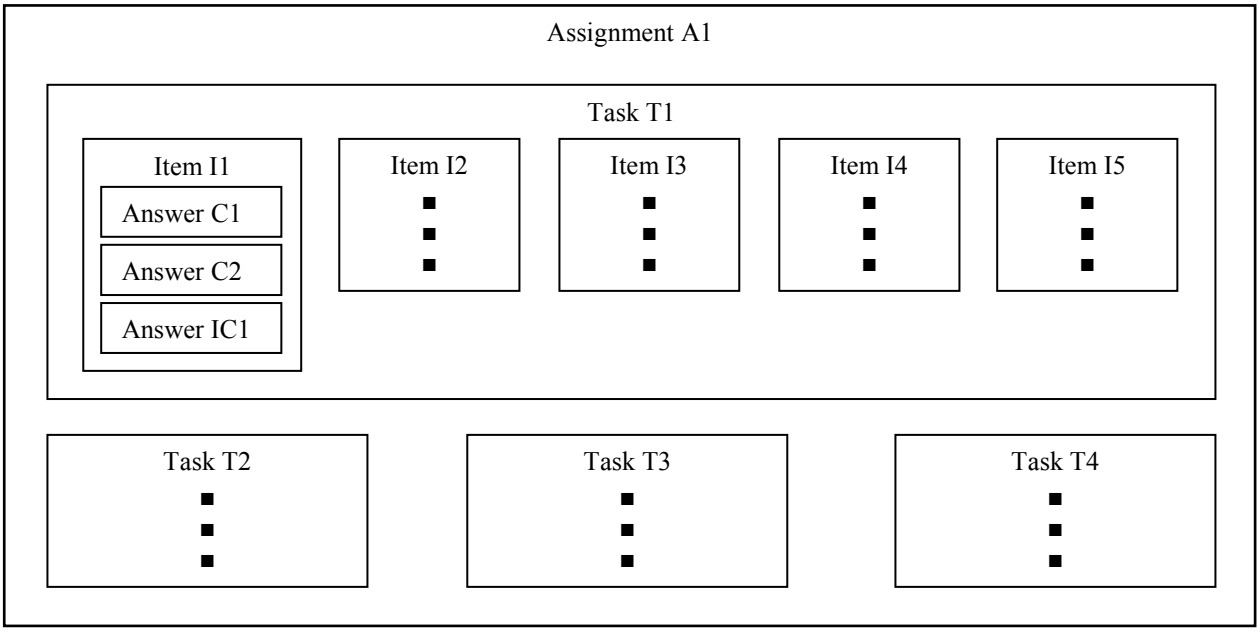


Figure 7: Abstraction of relationships shared by Assignment, Task, Item, and Answer objects within the AGLS

Iteration E2. The second iteration of the elaboration phase (E2) involved developing the designs that were the result of E1. The database would consist of not only tables and relationships, but also stored procedures used by the user interface and system logic. The database tables and relationships designed in E1 were created and filled with the required data for initial development. User interface screens were created based on sketches from E1. The stored procedures that would be required on these screens were created on an as-needed basis. Object classes from E1 were programmed and used within the user interface when

required. As an example, Figure 8 shows the structure of the Task object.

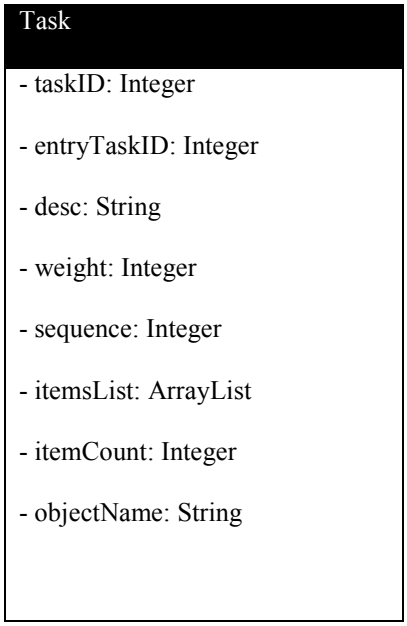


Figure 8: Task object class diagram

Testing was present throughout development in E2. Stored procedures were tested within the database itself and through data output from the user interface. Stored procedures were used for all database processes in order to increase speed and prevent SQL injection. Classes were built with a print function that allowed for easy output to test the contents and for more effective debugging and program tracing. The initial development of the database, user interface, and object classes marked the end of E2.

Iteration E3. The third iteration of the elaboration phase (E3) involved development of the backend grading that would be required for Access 2007 files. The first Intelligent Grading Module was created within this iteration. These modules were developed to provide the instructor-assisted grading services described in Section V. Since the structure of Access tables and queries are familiar (i.e. columns and rows), the interactive nature of the Grading Modules was not addressed during E3. The first step of the Grading Module development

was to read the required Tasks, Items, and Answers that were already established in the database. For the initial phases of development, these associations were entered directly into the database system instead of through the user interface. An array of Tasks was created and each Task object also contained an array of Items that had associated correct and incorrect Answers (see Figure 7). Correct answers can also contain partial credit or extra credit. Once these objects were established, files were ready to be graded.

Each file that is to be graded needed to be analyzed separately. Access files were read via several .NET connection technologies. An ADOX (ActiveX Data Object Extensions) Catalog was used to determine what can be done to each file. An ADODB (ActiveX Data Object Database) Connection was created to allow basic database functions to be performed on the Access 2007 file. In addition to these, an OLEDB (Object Linking and Embedding Database) Connection object was required to make inquiries for certain types of queries (i.e. parameter, wildcard, etc.). Table 5 summarizes the connection technologies that were utilized and their related uses. After the connections were established, several programmatic loops were implemented in order to perform the necessary grading. The first loop traverses through all Tasks that had been created. Within each Task, the individual Items were addressed. Every Item has its own associated Answers. When an Answer matched the students answer (as accessed through one of the above connection technologies), the Item was flagged as either correct or incorrect and dealt with accordingly. If any single Item failed within a Task, the Task was to be considered incorrect and appropriate feedback given. Once all feedback was accumulated, it was posted for review by the instructor. A completed cycle of grading all previously created Access Tasks, complete with adequate testing, marked the end of E3.

Iteration E4. The fourth iteration of the elaboration phase (E4) involved development of the backend grading that would be required for Excel 2007 files. This was the second

Intelligent Grading Module developed for the AGLS. Because of the vast number of possible correct and incorrect answers of an Excel file, the interactive features were included in E4. The process for collecting data to fill the Task, Item, and Answer objects was repeated from the Access Grading Module developed and tested in E3. Once established, files were ready to be graded.

Table 5: Access 2007 connection technologies and AGLS uses

Connection Type	Uses
ADOX Catalog	Provide access to tables, views, procedures and related fields, SQL, and details
ADODB Connection	Open and close files and create link to ADOX Catalog objects
OLEDB Connection	Run SQL against Access database files to allow manipulation

Just as with Access, each Excel file submitted was to be analyzed and graded independently. During this iteration, the Excel 2007 Object model was used to access elements within the file. Excel 2007 was loaded onto the Server and an instance of Excel was opened for each file being graded. Looping through Tasks, Items, and Answers was similar to the Access process developed in E3, but the process for dealing with answers that were neither correct nor incorrect was to be dealt with interactively. Once the AGLS found an "unknown" answer, the interactive grading module was called. The interactive grading module effectively used the knowledge provided by the instructor in order to "learn" various combinations of correct and incorrect answers and provide feedback (also defined by the instructor) for the student. Based on the position in grading (what Item and what Task), the AGLS was interrupted and the instructor was prompted. The prompt displayed what Item was causing a problem, what the suggested answer to the Item was, and what the student's

answer was (as found using the Excel 2007 Object model). Figure 9 shows an example of this interrupt. The instructor flagged an answer as correct, incorrect, partial credit, or extra credit and gave feedback accordingly. After the prompt was completed, the answer was stored in the database and grading started again at the beginning of the file that caused the interrupt. A completed cycle of interactively grading all previously created Excel Tasks, complete with adequate testing, marked the end of E4.

Iteration E5. The final iteration of the elaboration phase (E5) involved development of the user interface to be used for Assignment associations and Task entry. The modules developed in E5 fuel the success of the Grading Modules developed in E3 and E4. E5 consists of a series of web-accessible ASP.NET input screens for instructors.

Stored procedures for insertion of Tasks were built based on the Items and Answers required for each Task. When an instructor created a Task, s/he would supply the Key Answer to be used. The user interface needed to be adaptable in such a way that different Task requirements could all be addressed on the same screen. The stored procedures that were developed worked in such a way that only the necessary information was used to create the Answers associated with each Item. All data collected – whether relevant or not – could then be sent to the stored procedure as parameters. The stored procedure would determine what to use and what to ignore. Once initial (key) Answers were established and after grading had occurred, the instructors also needed a way to edit or delete certain Answers.

Once an Answer is found to be errant, the instructor can simply change the Answer or render it inactive. Rather than allowing Tasks and Item Answers to be directly edited, the option of "deactivating" a Task was given. Feedback for any Answer could be changed and type (correct or incorrect) could also be changed. This process is necessary in order to maintain the desired level of accuracy within the AGLS. The creation of Library Entries,

complete with gradable Tasks and associated Answers, marked the end of E5.

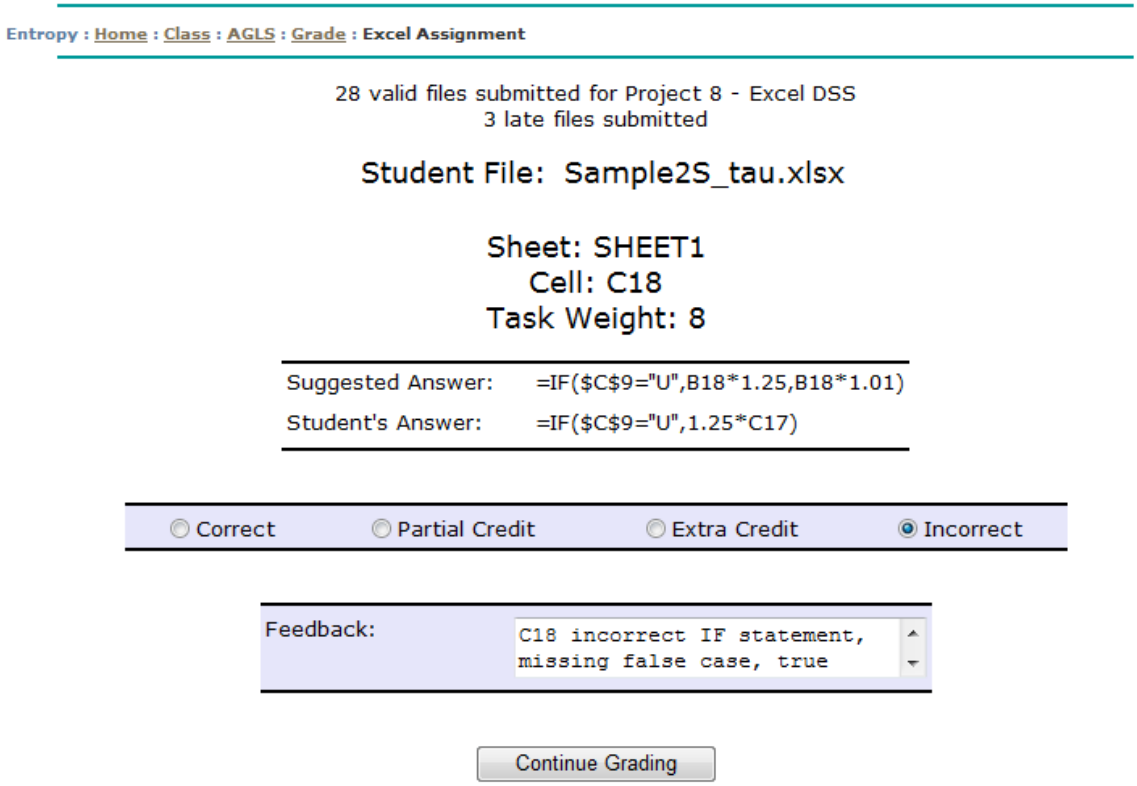


Figure 9: Instructor Prompt to Identify Student Answer as Correct or Incorrect and Associated Partial/Extra Credit and Feedback.

The completion of the elaboration phase enabled more focused pieces of the AGLS to be developed. The high-risk elements that held the key functionality of the system were completed in E1 – E5 and other elements were dealt with in the other phases of the Unified Process.

C. Construction Phase

The construction phase serves many purposes in a software development project (Satzinger et. al, 2005, p. 46-47). This phase involved a continuation of design and implementation of components in the AGLS, finalization of more predictive components such as the user interface, and the identification of limitations within the system. The construction phase consisted of five iterations which are denoted as C1, C2, C3, C4, and C5.

Iteration C1. The first iteration of the construction phase (C1) involved the development of a module that added functionality to both Intelligent Grading Modules (Access and Excel). The concept of Alternate Names was built to aid in grading. An Alternate Name is defined as an acceptable, but not ideal name for an Access or Excel Object. C1 included Alternate Names for Access Tables, Queries, and Field Names as well as Excel Sheets, Scenarios, and Solver Constraints. The problem addressed in C1 was how the system could manage multiple Alternate Names for all student files that were submitted. The solution involved another interrupt (similar to the Excel Interactive Grading interrupt) to the user/instructor. Figure 10 shows an example of this interrupt.

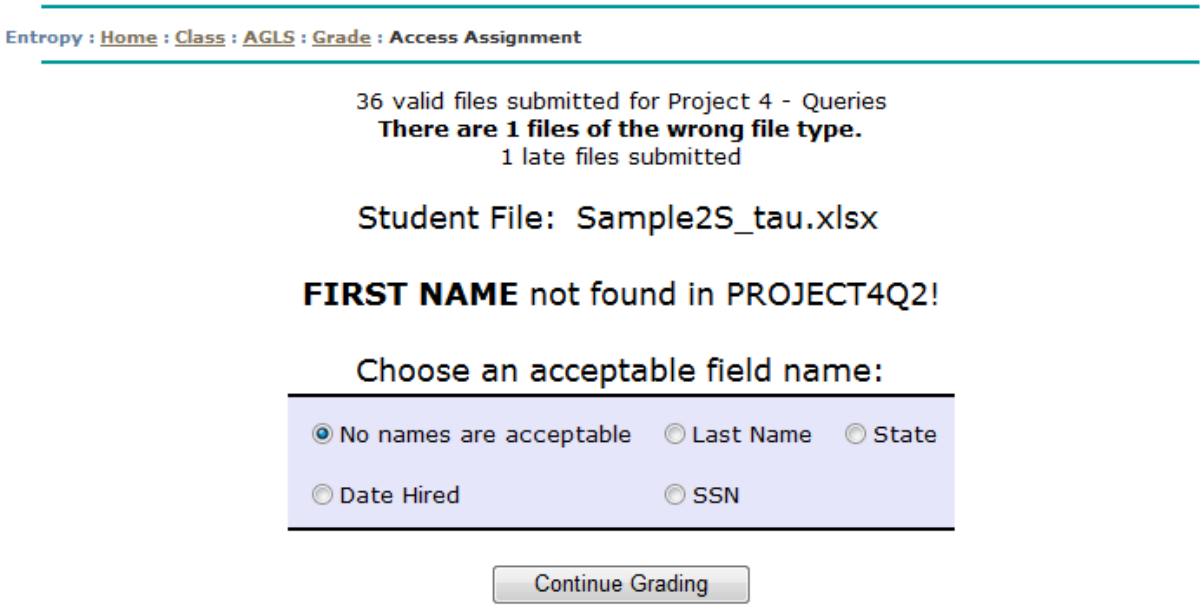


Figure 10: Instructor Prompt to Identify an Acceptable Alternate Name

When a student's file begins the grading process, the database is queried for the gradable Tasks, associated Items, and existing Answers. These are stored in an array of Task objects (reference E3 and E4 above). C1 added steps to the grading process that included querying for known Alternate Names (also stored in the database by Library Entry) and

writing newly selected Alternate Names to the database. When the user is prompted, Viewstate and Session variables are used to create copies of the Tasks array. Once an Alternate Name is selected, the Tasks array is altered to include the Alternate Name instead of the original key Answer given. Each student begins with the key Tasks array, but the Answers are changed based on the valid Alternate Names for the current Library Item. This method allows the AGLS to effectively manage different Tasks for each student based on his/her submitted file (and the aid of the user).

Each valid Alternate Name is stored in the database and associated with the given key Answer, the type of Object the name applies to (Table, Field, Sheet, etc.), and the Library Entry that the Alternate Name is valid for. Successful reading, writing, and use of the Alternate Names module in Access and Excel marked the end of C1.

Iteration C2. The second iteration of the construction phase (C2) involved the required completion of the ASP.NET front end user interface. Until this phase in development, the screens were simplistic and did not conform to one theme or design. The design and Cascading Style Sheets (CSS) used in Entropy allowed for some easy standardization of ASP.NET objects such as GridViews and DetailsViews that were used in the AGLS user interface, but some items, such as the Interactive Grading and Alternate Names modules needed to be formatted. These formatting additions were not time consuming but were necessary to deliver a quality system for deployment.

In addition to basic formatting and standardization of a design, C2 allowed for the flow of the user's experience to be finalized. The AGLS menu within Entropy was simplified to eliminate the confusion that could be caused by the initial design. Links within the Entropy system were added that informed users that the AGLS was available for Beta testing. The final step in C2 was to add some user direction to the essential screens that will be used by

instructors. Feedback from users led to constant updates to these messages throughout the remainder of development. Finalizing formatting issues, navigation links, and placement of informational messages marked the end of C2.

Iteration C3. The third iteration of the construction phase (C3) involved the development of strategies to provide possible Plagiarism Detection for student submitted Access and Excel files. Each Office product was given its own technique for Plagiarism Detection during C3.

For Access, instructors are expected to assign a gradable Task that simply records the creation time for one Table and one Query for each Library Entry. Once all files had been graded, an instructor could view a report that shows creation times for all Tables and Queries grouped by Library Entry. If matching creation times were found, the instructor was notified that these files were suspect and should be investigated for possible plagiarism.

For Excel, instructors are asked to provide a downloadable template to their students. These downloadable templates are embedded with the student's Entropy login ID. When the template is filled in and uploaded back into Entropy, the uploading student's ID is also known. When an instructor includes Plagiarism Detection in the grading process (an optional feature), then each file is tested. The testing process involves noting the uploader's ID (found by the filename of the uploaded file) and comparing this ID to the embedded ID. If a mismatch is found, the student's file is not graded and is flagged for review by the instructor.

Plagiarism Detection is not an infallible process, but does provide an additional level of detection required by instructors. The incorporation of Plagiarism Detection modules in both Access and Excel Grading Modules marked the end of C3.

Iteration C4. The fourth iteration of the construction phase (C4) involved the resolution of several issues that surfaced in the Access Grading Module. The ADOX and

ADODB objects used to read an Access file were found to have some inconsistencies that were not found in the initial testing done.

ADOX objects contain internal objects that represent Tables, Views, and Procedures. For the required initial needs of the AGLS, it was necessary to use all three of these objects. When the Grading Module was first developed (in E3), Procedures were not utilized. In Access, a Table is equivalent to an ADOX Table, and a normal Query is equivalent to an ADOX View. The inconsistency comes into play when a parameter is introduced into an Access Query. For example, if a query is written to display table records for a given week number and the week number is unknown at the time of writing the query, a parameter is needed. When parameters exist, a Query is no longer an ADOX View, but becomes an ADOX Procedure. In order to function properly, the backend SQL Command for Procedures needed to be analyzed. This was accessible and reprogrammed into the Access Grading Module.

In addition to Procedure adaptation, both ADOX Views and Procedures needed to be altered to allow for successful grading. Wildcards in Access Queries are represented by the "*" and "?". These represent "any number of characters" and "one character", respectively. The SQL that was being used by the AGLS needed to be altered to use "%" for Access's "*" and "_" for Access's "?". For each Query being graded, the backend SQL Command was read in as a String and the String's replace function was called to provide the needed manipulations.

The final error dealt with in C4 was the delay of processing files in the Access Grading Module. To narrow down where lags occurred, ASP.NET's Tracing was used. The benefit of this was the customized output that allowed easy identification of specific breakpoints and the time elapsed between them. For example, a specific lag of 27 seconds

was found and pinpointed to the line of execution that created the lag. After the lag sources were identified, the code was modified in order to eliminate lags. The 27 second lag example was reduced to a 1 second maximum execution time. The integration of ADOX Procedures, the modification of SQL Commands, and the successful elimination of major lags marked the end of C4.

Iteration C5. The last iteration of the construction phase (C5) involved a reworking of the processes used in the Excel Grading Module. It was necessary to develop a new Excel process that did not require opening excessive instances of Excel on the server. Two main reasons for this redevelopment were key in making this decision. One issue involved popup boxes within the student's file (normally the result of a circular reference). In addition, the memory needs of the server needed to be reduced. The Excel XML Grading Module was developed and implemented in C5.

The first task in developing the Excel XML Grading Module was familiarization with Microsoft's 2007 OpenXML schema. By renaming an .XLSX file to have a .ZIP extension, all of the native XML was exposed. Files were analyzed and relationships were noted and tested. Table 6 shows the relevant XML files that were used and their location within the Excel ZIP structure. The student's file needed to be copied (to a .ZIP file) and decompressed on the server. Once this had been completed, several files had to be analyzed. Because of the relationships that existed between the XML files and the extensive overhead that would be established by constantly reading the files, more object classes were created within the programming logic of the Excel Grading Module.

Several complex algorithms for reading and analyzing the XML files were developed and integrated into the Excel Grading Module. An example of such algorithms stemmed from the way Microsoft's OpenXML format handles formulas that are copied to adjacent cells.

When a formula is copied, Excel denotes this formula as a "shared" formula. The formula is given a shared ID and is stored in an XML node representing the first cell to use the formula. Any cell that uses the same formula records only the ID, not the newly created formula. In order for the AGLS to be able to read formulas, an algorithm was developed and implemented that copied the shared formulas to all cell objects that had the same shared ID, tokenized the formula, calculated the distance from the original cell, and built the new formula to be used. This process was further complicated because column references are alpha characters, not numbers. It was necessary to perform ASCII conversions and manipulations in this algorithm. Several algorithms such as this one were developed in order to parse the XML produced by Excel 2007 files.

Table 6: Microsoft Excel 2007 OpenXML Files Used by the AGLS

XML Location and Filename	Uses
..\xl\sharedStrings.xml	Contains an enumerated list of all Strings (non-numeric and non-formula cell entries) used throughout the Excel Sheets for the current file.
..\xl\workbook.xml	Contains an enumerated list of Sheets with names and associated files in the ..\xl\worksheets\ directory. Contains "defined names" that are used for AGLS grading of Excel Solver problems.
..\xl\charts*.xml	Each file represents one Chart or Object and its associated attributes and properties.
..\xl\styles.xml	Contains an enumerated list of all Fonts, Borders, Shadings, and Styles used throughout the Excel file.
..\xl\worksheets*.xml	Contains the actual contents of each Sheet in the Excel file. This file contains the actual Values and Formulas on a Cell-by-Cell basis. Contains all Scenarios built on each Sheet in the Excel file. Contains other Sheet attributes and properties such as page orientation.

Once the XML files had been read, the data for each student's file were stored in several Sheet and Cell objects. These objects contained massive amounts of data that was parsed from the XML files. Through the development and inclusion of the XML, the Grading Module was not only allowed to grade what had previously been included (reference E4), but also several of the initially desired gradable tasks identified in I1. Successful reprogramming of the Excel Grading Module to include XML parsing marked the end of C5.

The finalization of the construction phase marked the completion of all major components and modules of the AGLS. Even though this phase overlapped with the transition phase, the iterative and adaptive approach of the Unified Process allowed for successful development.

D. Transition Phase

The transition phase serves to provide beta testing and deployment so the declared benefits can begin from a software development project (Satzinger et. al, 2005, p.46-47). This phase involved final user feedback and acceptance, rollout of beta testing and bug elimination, and determining a support or maintenance plan. The transition phase consisted of three iterations which are denoted as T1, T2, and T3.

Iteration T1. The first iteration of the transition phase (T1) involved the creation of users in the system. Initially, only one user was involved with testing and grading. This individual was a graduate teaching assistant and the developer of the AGLS. The user was set up and used throughout the elaboration phase of development. Debugging and testing was the primary function of this user. Library Entries, Tasks, Items, and Answers were all created by this user and this user performed all preliminary grading in the AGLS. For the beginning iterations of the construction phase, two more users were created for testing. One of these users was an Information Systems faculty member and the other was an Operations

Management faculty member. These users created their own Assignments, Library Entries, Tasks, Items, and Answers. Once the major parts of the AGLS had been fully developed and tested by the three existing users, other users were created. These users included four new Information Systems instructors, one Computer Science instructor, and four more graduate Teaching Assistants. The successful integration and use of the AGLS by these multiple users marked the end of T1.

Iteration T2. The second iteration of the transition phase (T2) involved the training of instructors and other users of the AGLS. The main course of training was one-on-one sessions with the individuals who would be using the system for grading. Many times this training was coupled with a basic training of the Entropy environment and the capabilities and limitations of the systems within Entropy. Before the bulk addition of users and their adoption of the AGLS as expressed in T1, a training session was administered. This session was voluntary and provided a live and interactive demonstration of the AGLS. Questions that attendees had were answered and feedback was received to aid in the continued development. By the end of T2, anticipations into the 2009 usage of the AGLS were made. Operations Management, MBA, and Computer Science courses are expected to adopt the AGLS for use in 2009. A collaboration of instructors has also led to the collective effort to balance the burden of entering shared Library Entries into the system when new textbooks and Assignments are adopted. Successful training, either by one-on-one or group sessions marked the end of T2.

Iteration T3. The final iteration of the transition phase (T3) involved the development of a maintenance plan. After the completion of T1 and T2, users had already been setup and trained to use the system. Critical bugs and required immediate changes were dealt with in as little turnaround time as possible. These were classified as bugs that hindered or halted

successful grading of assignments. Some of these bugs, once classified, resulted in changes that affected other phases of development (i.e. C4 and C5). Other recommended or desired enhancements have been received from users. These non-crucial changes are to be dealt with between academic semesters by the development team. During the time of use (creation, submission, and grading of assignments), no unnecessary changes will be made to the system. The creating and releasing of this maintenance plan marked the end of T3.

All of the phases of the Unified Process were iterative and interactive. Just as the theory of the Unified Process stated, these phases did not occur entirely separated and each affected the outcome and progression of the others. The process was followed in order to ensure the successful and efficient development and implementation of the AGLS.

VIII. HYPOTHESIS DEVELOPMENT

To understand the benefits of the Adaptive Grading/Learning System (AGLS), an experiment to test the AGLS and its support of current theory was developed. Four specific hypotheses surrounding learning theory were developed. For each of these hypothesis, S_1 is the set of assignments where grading was performed without the AGLS and S_2 is the set of assignments where grading was performed with the AGLS.

These hypotheses are based on current learning theory concepts that show practice ("responding") and feedback ("reinforcement") are the events, given by Gagne et al. (1992), that are most directly connected to student success (Martin et. al., 2007). Data collection from the experiment is used to adequately support or reject these null hypotheses.

Test 1: Affect on Quantity of Feedback.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the amount of feedback provided to students.

H_1 ($\mu_1 - \mu_2 < 0$): The use of the AGLS will increase the amount of feedback provided to students.

Feedback quantity is measured by a count of characters used in comments given by instructors through course management software.

Test 2: Affect on Quality of Feedback.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the quality of feedback provided to students.

H_1 ($\mu_1 - \mu_2 < 0$): The use of the AGLS will increase the quality of feedback provided to students.

Feedback quality is measured with an evaluation of feedback by an expert panel of instructors. Measurements are made before and after implementation of the AGLS. As found

in Section II, Martin et. al. (2007) show that providing adequate feedback (quantity and quality) is directly related to student success. Sufficient feedback decreases the repetition of incorrect answers in the future and increases the probability of repeating correct responses (Philips et al., 1988).

Test 3: Affect on Response Time.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the amount of time for an assignment to be graded.

H_1 ($\mu_1 - \mu_2 > 0$): The use of the AGLS will decrease the amount of time for an assignment to be graded.

Response time is measured by the difference between the due date of an assignment and date a grade was issued by the instructor. A timely reinforcement of correct answers and identification of the faults in an incorrect answer are helpful to learners (Kulhavy, 1977).

Test 4: Affect on Number of Regrading/Errors/Inconsistencies.

H_0 ($percent_1 - percent_2 = 0$): The use of the AGLS will not affect the amount of grade changes made by instructors.

H_1 ($percent_1 - percent_2 > 0$): The use of the AGLS will decrease the amount of grade changes made by instructors.

To measure the change in required regrading, errors found, and inconsistencies, the following formula can be used to determine the percent of grades changed.

$$\frac{(\#\text{ of Entries in Log} - \#\text{ of Grades})}{\#\text{ of Entries in Log}} \times 100 = \%\text{ of Grades Changed}$$

Formula 1: Calculations to Determine Percent of Grades Changed.

X. EXPERIMENT

The setup of the experiment involved ten different sections of the same course. The set of MIS 213 courses taught during the Fall semester of 2007 served as the control (pre-test) set of the experiment. These courses implemented grading via traditional means; Access and Excel assignments were graded individually by hand. The set of MIS 213 courses taught during the Fall semester of 2008 served as the experimental set of the experiment. These courses implemented grading using the Adaptive Grading/Learning System (AGLS) as the primary method for grading Access and Excel assignments. The method of grading used for assignments was intended to be the only variable between the control and the experimental sets. Development of the AGLS began in the early Spring semester of 2008. Some assignments were graded preliminarily by the AGLS and some were not. Without a clear distinction of the grading techniques used, it was necessary to discard any data collected during the Spring semester of 2008.

A. Data Gathering

Data was gathered via three different means. Quantitative data came directly from the backend database of the AGLS. Qualitative data came from two sources; an anonymous online student survey and an expert panel of instructors.

Data for the quantitative tests was gathered for the experiment came from the database designed and implemented for use in Entropy. This database contains tables that describe assignments given during a course and the grades and comments received by students as well as the dates involved with grading. Queries were constructed that filtered the data of these tables based on the information needed to address the tests and hypotheses stated in Section IX.

Only Assignments that were candidates to be graded by the AGLS were used when

gathering data. Since Entropy requires the instructor to declare the type of file expected for each Assignment created, only Assignments that were declared as Access 2007 or Excel 2007 Assignments were of interest. These are currently the only Assignments that the AGLS can grade. In addition, Assignments given to MIS 213 sections were of interest because this course utilized Entropy during the semesters in question.

Once a valid list of Assignments was constructed, the list was divided into four parts. The use of database views in SQL Server allowed the tables to be merged and filtered based on the given specific requirements of Assignments. A view was constructed to keep Fall 2007 Access Assignments, one to keep Fall 2007 Excel Assignments, one to keep Fall 2008 Access Assignments, and one to keep Fall 2008 Excel Assignments. For each of the four views created, all grades and assignment details were collected. The fields of interest for the purposes of the experiment were student's grade, feedback received, count of characters in feedback, date due, date graded, and the response time (difference between date due and date graded).

A student survey and an expert panel were used to provide measurement for the quality of feedback. Student surveys were given to those students who received grades from the AGLS. The survey provided an understanding of student perceptions of feedback quantity, feedback quality, and response time. Only students in the experimental set of courses participated in the survey. The survey given to students contained three questions used for data analysis in this experiment: one for quantity of feedback, one for quality of feedback, and one for response time. The survey was intentionally kept brief in order to promote a higher response rate. Emails were delivered to 363 students and 92 responses were recorded and analyzed. The response rate was approximately 25.34%.

The expert panel was constructed involving instructors who had not used the AGLS.

This panel was given an online survey that showed feedback received by students enrolled in MIS 213 courses. Fifty feedback items were chosen from the control set and fifty feedback items were chosen from the experimental set. Half of these feedback items were related to Access assignments, the other half were related to Excel assignments. The panel members were instructed to assume that each assignment graded had errors, all errors were found, and the resulting feedback addressed those errors. All items were compiled together so the members of the panel were unaware of which set they belonged to. The expert panel ranked the sample feedback as "Very Ineffective," "Ineffective," "Neutral," "Effective," or "Very Effective."

For the purposes of the expert panel, the four views were filtered further. All grades that had blank comments were eliminated from the lists because these comments would not provide adequate ranking. For this study, the focus was determining the quality of comments with a length greater than zero. In a sampling of data that excluded assignments that received a perfect score, in Fall 2007 introductory classes, 130 of the 429 graded assignments (30.30%) had no related comments from the instructor. For the similar sampling from data in Fall 2008 introductory classes, only 18 of the 1823 graded assignments (0.98%) had no related comments.

From the list of grades that received a comment, any grade of 0 (no credit) or 100 (full credit) was eliminated. Generally, these comments were generic enough to be eliminated. Normally a grade of 0 denoted a problem and was accompanied by either "Please see instructor" or "No submission." Grades of a 100 were given feedback that would read "Excellent work" or "Great job." These are inadequate for the purposes of the feedback quality test. All other feedback for grades was deemed valid for expert panel review. To compile the survey for the panel members, a random sampling was taken from the

populations of feedback in order to provide only 50 from each set.

In addition to the given data collection for feedback and response times, it was also necessary to determine how many entries were made to Entropy's grade log. Hypothesis 4 (see Section IX) addresses the issue of grading errors. By collecting the count of grades that were received by students and collecting a count of how many entries were made to the grade log, the percentage of grades in the log that were changed could be determined by the Formula 1.

B. Results and Analysis

Results and analysis was performed on a test by test basis. Each test is briefly summarized below and the results and analysis are given. More information about each test can be found in Section IX.

Test 1: Affect on Quantity of Feedback.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the amount of feedback provided to students.

H_1 ($\mu_1 - \mu_2 < 0$): The use of the AGLS will increase the amount of feedback provided to students.

Feedback quantity was measured by a count of characters used in comments given by instructors through Entropy. In order to analyze the results of this test, a t-test was performed assuming equal variance. Table 7 details the statistics gathered from the t-test of character count data. The p-value given from the t-test is less than $\alpha = 0.05$. With a p-value less than 0.05, there exists at least a 95% confidence level that the two populations are not the same, thus $\mu_1 - \mu_2 \neq 0$. This data supports a rejection of the null hypothesis (H_0). The use of the AGLS will affect the amount of feedback provided to students. Further analysis supports accepting the alternate hypothesis (H_1). Since the t statistic value is negative, the conclusion

that $\mu_1 - \mu_2 < 0$ can be supported.

Conclusion 1: The use of the AGLS will increase the amount of feedback provided to students.

Table 7: T-test Data Related to Character Count of Feedback

	Control Set	Experimental Set
Mean	45.10	71.36
Variance	4104.20	9213.33
Standard Deviation	64.06	95.99
Sample Size	628	3138
Hypothesized Mean Difference	0	
Degrees of Freedom	3764	
t Statistic	-6.57	
Two-tail p-value	5.83212×10^{-11}	

The survey responses related to the student's perceptions of the quantity of feedback from the AGLS showed that 62% either agreed or strongly agreed that the system provided a "sufficient (amount of) feedback." Thirty-eight percent strongly disagreed, disagreed, or were neutral in their opinion of feedback quantity.

Test 2: Affect on Quality of Feedback.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the quality of feedback provided to students.

H_1 ($\mu_1 - \mu_2 < 0$): The use of the AGLS will increase the quality of feedback provided to students.

Feedback quality was measured with an evaluation of feedback by an expert panel of instructors. Feedback was ranked by members of the expert panel as "Very Ineffective",

"Ineffective", "Neither", "Effective", or "Very Effective." The answers were ranked with a numerical value 1, 2, 3, 4, or 5 respectively. The data used in the test was the average of these numerical values. Table 8 details the statistics gathered from the t-test of expert panel reviews of feedback. With a p-value greater than 0.05, there exist at least a 95% confidence level that the two populations are the same, thus $\mu_1 - \mu_2 = 0$. This data supports acceptance of the null hypothesis (H_0). The use of the AGLS will not affect the quality of feedback provided to students. Since the populations are the same, further analysis cannot be performed to accept or reject the alternate hypothesis (H_1).

Table 8: T-test Data Related to Expert Panel Review of Feedback

	Control Set	Experimental Set
Mean	3.27	3.45
Variance	0.87	0.49
Standard Deviation	0.93	0.70
Sample Size	49	51
Hypothesized Mean Difference	0	
Degrees of Freedom	98	
t Statistic	-1.13	
Two-tail p-value	0.261440502	

Conclusion 2: The use of the AGLS will not affect the quality of feedback provided to students.

The survey responses related to the student's perceptions of the quality of feedback from the AGLS showed that 66% either agreed or strongly agreed that the system provided "clear and consistent feedback." Thirty-four percent strongly disagreed, disagreed, or were neutral in their opinion of feedback quality.

Test 3: Affect on Response Time.

H_0 ($\mu_1 - \mu_2 = 0$): The use of the AGLS will not affect the amount of time for an assignment to be graded.

H_1 ($\mu_1 - \mu_2 > 0$): The use of the AGLS will decrease the amount of time for an assignment to be graded.

Response time was measured by the difference between the due date of an assignment and date a grade was issued by the instructor. Table 9 details the statistics gathered from the t-test of response time data. With a p-value less than 0.05, there exists at least a 95% confidence level that the two populations are not the same, thus $\mu_1 - \mu_2 \neq 0$. This data supports a rejection of the null hypothesis (H_0). The use of the AGLS will affect the amount of time for an assignment to be graded. Further analysis supports accepting the alternate hypothesis (H_1). Since the t statistic value is positive, the conclusion that $\mu_1 - \mu_2 > 0$ can be supported (μ_1 is the mean of the control set and μ_2 is the mean of the experimental set).

Conclusion 3: The use of the AGLS will decrease the amount of time for an assignment to be graded.

Table 9: T-test Data Related to Response Time of Grading

	Control Set	Experimental Set
Mean	28.59	8.13
Variance	380.11	159.33
Standard Deviation	19.50	12.62
Sample Size	628	3138
Hypothesized Mean Difference	0	
Degrees of Freedom	3764	
t Statistic	33.43	
Two-tail p-value	9.7353×10^{-215}	

The survey responses related to the student's perceptions of the response time of feedback from the AGLS showed that 80% either agreed or strongly agreed that the system provided feedback "in a timely manner". Twenty percent strongly disagreed, disagreed, or were neutral in their opinion of feedback response time.

Test 4: Affect on Number of Regrading/Errors/Inconsistencies.

H_0 ($percent_1 - percent_2 = 0$): The use of the AGLS will not affect the amount of grade changes made by instructors.

H_1 ($percent_1 - percent_2 > 0$): The use of the AGLS will decrease the amount of grade changes made by instructors.

Table 10 gives the details used to calculate the percent of grades changed as well as the calculated percent. Since $percent_1 - percent_2 \neq 0$, the null hypothesis ($H_0$) can be rejected. The use of the AGLS will affect the amount of grade changes made by instructors. The conclusion can also be made that $percent_1 - percent_2 > 0$. This leads to a rejection of the alternate hypothesis (H_1) as well.

Conclusion 4: The use of the AGLS will increase the amount of grade changes made by instructors.

Table 10: Data Related to the Percent of Grades Changed

	Control Set	Experimental Set
Number in Grade Log	917	6067
Number of Grades Recorded	651	3133
Percent of Grades Changed	20.01%	48.36%

The high number of changes during the experimental set can be attributed to system development and changes. Because of additions and modifications of the AGLS during this

semester, many instructors regarded assignments multiple times. The regarding process changes all grades for the assignment being regarded.

XI. TECHNOLOGIES USED, SKILLS LEARNED, AND BENEFICIAL CLASSES

During the development of the Adaptive Grading/Learning System (AGLS), the skill sets of the following areas were actively improved. Each item includes a description of specific skills learned and beneficial courses taken during Graduate Studies.

A. Logic & Presentation Layer

All modules of the AGLS were developed in ASP.NET with a VB.NET codebase. Extensive use of Objects and Classes was required during the implementation of the designs created for the AGLS. Methods of communication between different web pages developed in ASP.NET were required. Alternate languages that could have been used included Java Server Pages, PHP, or C#.NET. ASP.NET and VB.NET were chosen because the AGLS developed was to be fully integrated with Entropy which was developed using ASP.NET and a VB.NET codebase. Style sheets and valid XHTML coding was required in order to provide for cross-browser support for the AGLS. Since the system was integrated into Entropy, the format and styles were required to be consistent.

B. Database Layer

The backend database for the AGLS was designed and developed using SQL Server 2005. MIS 555 (Database Management Systems) was essential in establishing the fundamentals required to develop a fully relational database. Effective management of Tables and Stored Procedures was necessary for successful deployment of the AGLS. Stored Procedures were used exclusively to access data within the database. This approach allowed a much more efficient and speedy execution than in-code SQL calls. MySQL was considered, but because of the Entropy integration requirements, SQL Server 2005 was chosen for the AGLS development. This decision allowed all AGLS tables and stored procedures to be migrated along with Entropy to SQL Server 2008.

C. System Analysis

The Unified Process was the software development methodology adopted for the AGLS development. CSC 550 (Software Engineering) and MIS 565 (Analysis, Modeling, and Design) were both beneficial in their review of the many methodologies available as well as their strengths and weaknesses. The ability to follow the selected process and identify the different phases and iterations within each phase was a valuable product of the AGLS development.

D. Project Management

The AGLS development required many project management skills. Management of faculty and graduate assistants occurred throughout the lifetime of the system's development. POM 572 (Project Management) gave the foundations required for successful management of this project.

E. XML Objects and Manipulation

The understanding of XML was essential to the redevelopment of the Excel Grading Module within the AGLS. Incorporating the VB.NET related objects and their manipulation as well as adapting new objects to store data is applicable in many real-world business applications. For example, the knowledge learned from XML uses explored during the development of the AGLS has already been used to produce output for a local Proposal Generation System (PGS). For the PGS, an output template was developed in Word 2007 and the XML Objects created were manipulated in order to produce the desired results.

XII. LIMITATIONS AND FUTURE RESEARCH

As with any experiment, there were several research limitations that effected the hypothesis development of this study. There was only one semester of data collected while using the developed AGLS. This limited data supply prohibited the ability to monitor if there was an increase of assignments. Learning theory suggest that in order to increase learning, an increase in real world assignments is required (Rakes, 1996). With only one semester included in the research, no comparison could be made. Without a comparison, an increase is impossible to measure. In addition to the inability to measure the number of assignments given, only one semester of data also prevented measurement of grade improvements.

In addition to only one semester of data, the AGLS was tested in only one subject matter. The courses involved were introductory courses that teach Access 2007 and Excel 2007. Although these courses are within the original needs expressed, they only allow testing of one group of students that could receive benefits from the system. Unfortunately with the research limitations expressed, a valid Web Service could not be developed. This Web Service would allow an instructor to bypass the interactive grading process. This would only be suggested when an adequate grading template has been refined to include the majority of expected answers. Each of these answers would be paired with a correct or incorrect status and appropriate feedback if needed. This feature of the AGLS would ideally be offered as a Web Service in which assignments can be sent to the system and each is graded without any required review from the instructor. Responses not identified as correct could either be flagged for future review or initially considered incorrect during this type of grading. Online courses could benefit from the availability of this feature. An example where this Web Service may be useful is when an instructor uses the AGLS to allow students to upload a small practice exercise. These smaller problems can be immediately graded and feedback

sent to the student and instructor to review. In order to create an effective Web Service portion of the AGLS, an extensively refined library of correct and incorrect answers would need to be associated to all Library Entries included in the Web Service. As the AGLS is utilized the Library will evolve and this enhancement can be addressed.

To manage the research limitations that came from the AGLS development, the AGLS should be incorporated into more courses for a few more semesters. Future research can be conducted once this has occurred. When more data has been collected, variables can be reduced and the comparisons can be made that will test questions such as how the number of assignments given to students has changed and if grades improve after the implementations of the AGLS.

Aside from research limitations, a few grading limitations surfaced during the development of the AGLS. Table 11 lists the limitations and their associated Grading Module (Access or Excel). During continual development and additions made to the AGLS, these problems will be addressed and incorporated into the system. As feedback is received, new limitations will be addressed according to their severity.

Table 11: AGLS Grading Limitations and Their Associated Grading Modules

Grading Module	Limitation
Access	"ID" cannot be the Primary Key of a Table being tested in the AGLS.
Access	Cannot test Field Type for a Query.
Access	Forms cannot be accessed and graded.
Access	Reports cannot be accessed and graded.
Access	Formatting cannot be graded.
Excel	Cannot test a Date as a Cell Value.
Excel	Cell values can only range from A1 to Z99.

XIII. IMPLEMENTATION AND EXPERIMENT TIMELINE

Table 12 shows the major activities that occurred during the development of the AGLS. The proposed planned timeline is given along with the actual timeline for development. These activities are also categorized by phase and iteration of the Unified Process.

Table 12: Prospective vs. Actual Timeline for the AGLS Implementation and Experiment

Activity	UP Phase	Planned	Actual
Learning theory literary review	I1	Fall 2007	Fall 2007
Identifying system benefits	I1	Fall 2007	Fall 2007
Defining system need	I1, E1	Fall 2007	Fall 2007
Faculty interviews	I1	Fall 2007	Fall 2007
Defining scope	I1	Fall 2007	Fall 2007
Determine requirements	I1	Fall 2007	Fall 2007
Database design (tables/relationships)	E1	Dec 07	Dec 07 - Jan 08
Initial user interface design	E1	Dec 07 - Feb 08	Dec 07 - Feb 08
Object class designs	E1	Jan - Feb 08	Jan, Sept - Oct 08
Database development (tables/relationships)	E2	Dec 07 - Mar 08	Dec 07 - Oct 08
User interface programming	E2, E5, C2	Dec 07 - Feb 08	Dec 07 Jan, May - Aug 08
Object class development	E2	Jan - Feb 08	Jan, Sept - Oct 08
Access 2007 Grading Module development	E3, C4	Jan - Mar 08	Feb - Apr, Jul - Sept 08
Excel 2007 Grading Module development	E4, C5	Feb - Apr 08	Mar - May, Aug - Oct 08

Activity	UP Phase	Planned	Actual
Alternate names module development	C1	Unplanned	Apr - Jun 08
Plagiarism detection module development	C3	Feb - Mar 08	Feb - Mar, Jul - Sept 08
User creation & addition	T1	Jan - Apr 08	Jan - Oct 08
One-on-one training of users	T2	Mar - Apr 08	Apr - Oct 08
Group training session	T2	Mar - Apr 08	Jul 08
Maintenance plan development	T3	Mar - Apr 08	Jun - Sept 08

XIV. CONCLUSIONS

Development of a system based on pedagogy has been shown to affect the quantity of feedback received and the response time of the feedback in a positive manner. Following the Unified Process methodology provided a framework that enabled successful implementation of the pedagogical objectives as related to grading and feedback. Specifically, the Unified Process supported the modular development of the Adaptive Grading/Learning System.

Feedback has been identified as a key component of student success. This builds upon the research from the cognitive, behavioral, and resource-based views of learning. To understand the impact of the AGLS on feedback, four hypotheses were created and experiments were developed to test them. Three null hypotheses were rejected and one was accepted. Specifically, the data analysis from the four tests performed yielded the following results:

- The use of the AGLS will increase the quantity of feedback provided
- The use of the AGLS will not affect the quality of feedback provided
- The use of the AGLS will decrease the amount of time for an assignment to be graded.
- The use of the AGLS will increase the amount of grade changes made by instructors.

REFERENCES

Course Technology - CaseGrader: Microsoft Office Excel 2007 Casebook with Autograding Technology. (Online). Available: http://www.course.com/catalog/product.cfm?isbn=978-1-4239-9823-5&CFID=337118&CFTOKEN=65583638 [October 12, 2007].

Course Technology - SAM 2007 Training V1.0. (Online). Available: http://www.course.com/catalog/product.cfm?isbn=978-1-4239-1305-4 [October 13, 2007].

Crews, Thad and Murphy, C. (2008). *CASEGRADER: Microsoft Office Excel 2007 Casebook with Autograding Technology*. Boston, MA: Thomson Course Technology.

Dewald, N. (1999). Web-based Library Instruction: What Is Good Pedagogy? *Information Technology and Libraries, Chicago, 18 (1)*, 26.

EMC/Paradigm: College Catalog: Computer Technology: Snap Web-based Training & Assessment: Snap 2007. (Online). Available: http://www.emcp.com/product_catalog/index.php?GroupID=1926 [October 13, 2007].

Gagne, R. (1965). *The Conditions of Learning* (1st Ed.). New York: Holt, Rinehart & Winston.

Gagne, R. (1985). *The Conditions of Learning* (4th Ed.). New York: Holt, Rinehart & Winston.

Gagne, R.M. (1988). Mastery Learning and Instructional Design. *Performance Improvement Quarterly, 1 (1),* 7-18.

Gagne, R., Briggs, L. & Wager, W. (1992). *Principles of Instructional Design* (4th Ed.). Fort Worth, TX: HBJ College Publishers.

Kay, David G. (1998). "Large Introductory Computer Science Classes: Strategies for Effective Course Management," *ACM SIGCSE Bulletin, 30*, 131-134.

Kulhavy, R. W. (1977). Feedback in written instruction. *Review of Educational Research, 47 (1)*, 211-232.

Martin, F., Klein, J. & Sullivan, H. (2007). The Impact of Instructional Elements in Computer-Based Instruction. *British Journal of Educational Technology, 38 (4)*, 623 - 636.

Miller, P. et. al. (1981). "Plagiarism in computer sciences courses (Panel Discussion)," *Proceedings of the twelfth SIGCSE technical symposium on Computer science education*, 26-27.

National Research Council (1999). *BeFIT: Being Fluent in Information Technology*. Washington, DC: National Academy Press.

Philips, T., Hannafin, M., & Tripp, S. (1988). The effects of practice and orienting activities on learning from interactive video. *Educational Communication and Technology, 36*, 93-102.

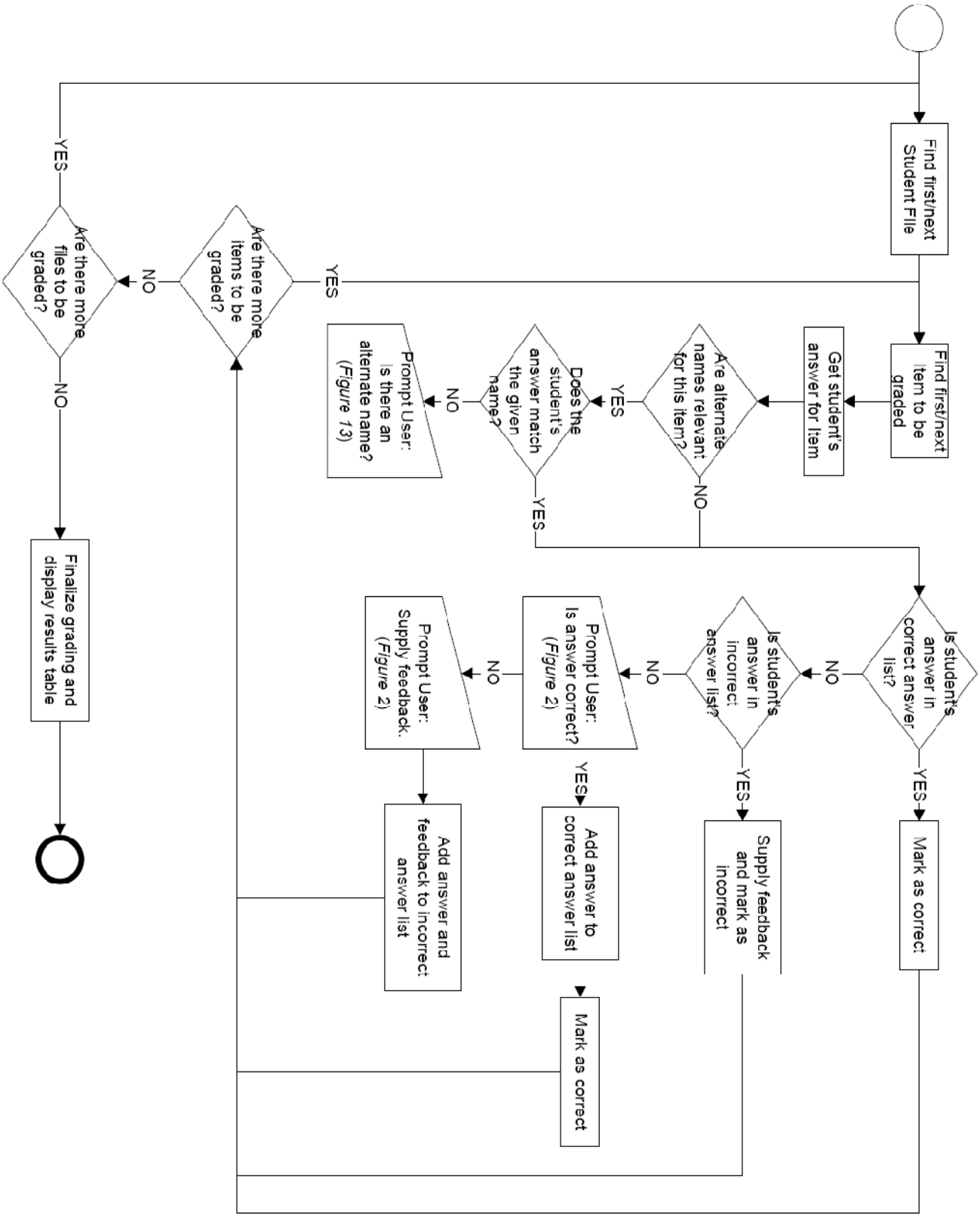
Rakes, G. (1996). "Using the Internet as a tool in resource based learning environment". *Educational Technology, 6 (2)*, 52-29.

Reiser, R. A. & Dick, W. (1996). *Instructional Planning: A guide for teachers* (2nd Ed.). Allyn and Bacon Publication.

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2005). *Object-Oriented Analysis and Design with the Unified Process*. Boston, MA: Thomson Course Technology.

University of North Carolina Wilmington. (2007). *Undergraduate Catalogue 2007-2008*. (Online). Available: http://uncw.edu/catalogue/documents/catalogue.pdf [September 24, 2007].

APPENDIX A: Process Flow of Adaptive Grading in the AGLS



APPENDIX B: Gradable Tasks for Access 2007

Table names

Field attributes (name, data type, default value, primary key, etc.)

Data entered into tables

Query names

Fields shown in queries

Criteria (=, < >, <. >, < =, > =, LIKE, *, %, BETWEEN, AND, OR)

Parameter Queries (using [ ])

Calculated columns/aggregate functions (SUM, COUNT, AVERAGE, GROUP BY, etc.)

Sorting for reports

Properties for fields of a report

APPENDIX C: Gradable Tasks for Excel 2007

Static Cell contents (strings, numbers, etc.)

Cell text formatting (alignment, bold, italics, underline, font face, type, etc.)

Cell type (percent, currency, text, decimal places, etc.)

Sheet formatting (page orientation, etc.)

Formulas (basic arithmetic, functions (SUM, MIN, MAX, etc), IF statements)

Cell references (absolute, relative, combination/partially absolute)

Charts (title, chart type, etc.)

Sheet names (Sheet1, Sheet2, Answer Report, etc.)

Scenario Manager (scenario summary, changing cell, scenario names, scenario values, result cells, scenario result values)

Solver (default values, constraints, target cell, adjustable cells, answer report, extension cases, answer report answers, final value answers)

APPENDIX D: Database Design for the AGLS