ATHABASCA UNIVERSITY

STUDENT ATTRITION IN COMPUTER SCIENCE COURSES: A COMPUTATIONAL PERSPECTIVE

BY

STEVEN KALMAR

An essay submitted in partial fulfillment
Of the requirement for the degree of
MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta
August, 2013

© Steven Kalmar, 2013
DEDICATION

I dedicate this essay to my wife Allison for her love, support and encouragement and my children Aidan, Hannah and Lillian for giving me the time to complete this work.
ACKNOWLEDGEMENTS

I would to thank Dr. Vive Kumar for his time and effort.
Table of Contents

Abstract .............................................................................................................................................. 6
1 Introduction ........................................................................................................................................ 6
2 Attrition and Failure in Computer Science ..................................................................................... 7
   2.1 Computer Science Attrition Rate vs. Overall Attrition Rate .................................................... 7
   2.2 Attrition rate for distance education computer science courses ............................................. 7
   2.3 Attrition rate by year in post-secondary education ................................................................. 8
   2.4 Attrition rate comparison between full and part time students ............................................. 8
   2.5 Factors affecting attrition rates ............................................................................................... 8
      2.5.1 Poorly Designed Introductory (CS1) Lab Courses ............................................................. 9
      2.5.2 Lack of practice and feedback ......................................................................................... 10
3 Feedback ......................................................................................................................................... 10
   3.1 Introduction to Feedback ....................................................................................................... 10
      3.1.1 Purpose of Feedback ....................................................................................................... 10
      3.1.2 Time to Provide Feedback .............................................................................................. 10
      3.1.3 Seven Principles of Good Feedback .............................................................................. 11
   3.2 Types of Feedback .................................................................................................................... 12
      3.2.1 Summative Feedback ...................................................................................................... 13
      3.2.2 Formative Feedback ...................................................................................................... 13
      3.2.3 Concurrent Evaluation .................................................................................................... 14
      3.2.4 Ipsative Feedback .......................................................................................................... 14
      3.2.5 Affirmative Feedback .................................................................................................... 15
      3.2.6 Corrective Feedback ...................................................................................................... 15
   3.3 Timeliness of Feedback .......................................................................................................... 16
      3.3.1 The Changing Timeliness ............................................................................................... 16
      3.3.2 How Long is Too Long .................................................................................................. 17
4 Personality Types and Learning Styles ......................................................................................... 17
   4.1 Personality Types ..................................................................................................................... 17
      4.1.1 Myers-Briggs Type Indicator ......................................................................................... 17
      4.1.2 What Personality Types Suit Computer Science Students ........................................... 18
   4.2 Learning Styles ....................................................................................................................... 19
Abstract
In this paper the issues surrounding attrition in computer science and related disciplines, the nature of feedback and how it helps to improve student comprehension, personality types & learning styles and intelligent tutoring systems will be covered through a comprehensive literature review. Based on this review, a system is proposed that, continuously and formatively, gathers data from students while they are completing their course work. The system stores that data and then analyzes that data to determine the course competencies the students have yet to achieve as well as identifies gaps in their knowledge and study skills. The analysis could be used to provide high quality, timely, interactive and proactive feedback to students as they continue to complete their course work. This paper concludes with a discussion on challenges facing the proposed feedback system and on how the feedback system would benefit students and instructors with tackling attrition.

1 Introduction
Attrition in computing disciplines is generally accepted as a critical issue that needs supplementary examination (Beaubouef & Mason, 2005; Lasserre & Szostak, 2011). The attrition rate of computer science degrees is higher than average attrition rate in other disciplines, by quite a margin (Beaubouef & Mason, 2005; Lasserre & Szostak, 2011). In addition, the dropout/failure rates in introductory courses are higher than the average dropout/failure rates for advanced courses (Statistic Canada, 2009). It stands to reason that if students do not do well in the introductory programming course it will make it more likely that they will not continue to pursue a degree in the discipline. There are number of factors that have been identified as contributing to the high failure and dropout rates in introductory programming courses. This paper will propose a solution that would help address poor feedback and lack of practice, two important factors related to attrition.

The paper will start with a look at attrition in computing related disciplines, including failure rates for introductory computing courses. After examining attrition the paper will discuss feedback in a general sense followed by looking at some of the different types of feedback that can be given to students. This will be followed by a discussion of other issues related to feedback such as when and how to give feedback. As personalized feedback is important to providing high quality feedback, the detailed look at personality types, based on the Briggs Meyers type indicator, including examining what personality types are common in computing related disciplines. The discussion on personality types will be followed by looking at learning styles, using the Felder-Silverman and Kolb learning style models. This will lead into an examination of the different levels of competence, using Bloom's Taxonomy. The taxonomy can be used to show what course competencies have been achieved by students and present this information to them using a dashboard. Knowledge of what level of competence a student is expected to achieve is important since comparing the expectations of student competence with the competence the students have currently achieved is the first step to providing appropriate feedback. The paper will then discuss intelligent tutoring systems and some of the feedback mechanisms arising out of this area. The section on intelligent tutoring system will also outline the four major components of an intelligent tutoring system. The conclusion will include an outline of the proposed system which will provide high quality, timely feedback to students. Included in the proposal will be details of how the proposed system
will alter existing programming workflow. The conclusion will also contain a discussion on further research required to deal with the problem of program attrition and course dropout rates and some of the challenges that might be faced with the proposed system.

2 Attrition and Failure in Computer Science
Attrition of students in post-secondary studies is a fact of life for post-secondary institutions (Willcoxson, Cotter, & Joy, 2011). Attrition is particularly pronounced in computer science and other technical programs. This section will start with a brief on attrition rates in computer science, and then it will look at attrition rates in distance education computer science courses compared to traditional brick and mortar computer science courses. The section will also examine how the year of study affects the attrition rate as well as the impact of part-time study versus full time study. The section will conclude with a discussion on factors that affect attrition and the reasons for non-completion of post-secondary education.

2.1 Computer Science Attrition Rate vs. Overall Attrition Rate
Computer science programs have a history of problems both in recruiting students and keeping students in the program. The attrition rate in computer science, specifically the introductory courses has been found to be 30%-50% according to Lasserre & Szostak (2011, p. 133) and between 30% and 40% according to Beaubouef & Mason (2005). Beaubouef & Mason (2005) noted that their personal observations have attrition rates as high as 60%. A Statistics Canada (2009) study of attrition in the Atlantic provinces reports that the number of students leaving post-secondary education in their first year of study is around 15%, and the rate of students, in their first year, changing programs and/or universities is 5% (Statistic Canada, 2009). Another interesting piece of information to note in the Statistics Canada (2009) report is that out of the 15% of students who left post-secondary education, nearly 25% of them returned to some post-secondary institutions, including some of these students even returning to their previous program of study (Statistic Canada, 2009), meaning that some of the attrition rate information may be artificially high if this additional information of returning students is not taken into account. The attrition rates for computer science are even worse in the context of the attrition rates for women and minorities (Sloan & Troy, 2008; Corney, Teague, & Thomas, 2010).

These figures clearly indicate that while a certain level of attrition is to be expected by universities, the rates for computer science and other technical disciplines are well above the average attrition levels across all disciplines. The high attrition rates have led to a variety of research to help determine the factors that have the largest impact on the attrition and to determine strategies that should be employed to help remedy the situation.

2.2 Attrition rate for distance education computer science courses
The number of institutions that are offering distance education courses has increased from 56% of post-secondary institutions in 2000-2001 (Waits & Lewis, 2003, p. 4) to 66% by 2006-2007 (Parsad & Lewis, 2008, p. 5). The numbers are even more impressive when looking at enrollment in online courses. The rate of enrollment in online courses has increased from 9.6% in 2002 to 31.3% in 2010 (Allen & Seaman,
This indicates that more and more courses and even entire degrees are being or will be offered in an online manner.

The trend of increasing distance education enrollment coupled with the fact that the attrition rate among distance education courses is 10%-20% higher than in traditionally taught courses (Angelino, Williams, & Natvig, 2007, p. 5), implies that computer science with its already higher than average rate of student attrition needs to ensure that their distance education offerings take appropriate steps to meet the needs of distance education students.

2.3 Attrition rate by year in post-secondary education

In comparison to the 15% figure quoted earlier for first year students' attrition rate, Statistics Canada (2009) reports that attrition rates in second year dropped marginally compared to first year, from 15% to fewer than 12%. However, these rates when taken together imply that on average 25% of all university students have left their studies by the end of second year. In computer science, with its higher than average attrition rate, approximately double the average, means that computer science has a cumulative attrition rate of nearly 50% after the second year of study.

Since the greatest rate of attrition is in the first year of study the greatest effort should be directed towards improving introductory course offerings with the goal of reducing the dropout rate for these introductory courses and the attrition rates for the first half of computer science programs.

2.4 Attrition rate comparison between full and part time students

It is also important to look at the differences in attrition between full and part time students. Many more students are now working part time while studying to earn their degrees. According to Mary Jane Feldman (1993), retention rates of part-time students are lower than those of full time students. Given the increasing tuition rates at many post-secondary institutions more students are taking studies on a part-time basis. This increase in part-time enrollment coupled with a higher dropout rate among part-time students will have an adverse effect on retention in computing related disciplines, which already have high attrition rates. The implication of this is that care should be taken to ensure that students, especially those who are part-time students, have access to resources when they are able to study, such as instructors and instructional assistants, who are able to provide feedback on problems the students may be having in a timely manner.

2.5 Factors affecting attrition rates

There are various factors which may affect the attrition rate for computer science students beyond those factors which contribute to the general attrition rate at a post-secondary institution. Beaubouef and Mason (2005) provide seven factors which they believe contribute to the high rate of attrition among computer science students. The seven factors which have an impact on attrition for computer science students are:
1. Poor Advising Before and During College
2. Poor Math and Problem Solving Skills
3. Poorly designed CS1 Lab Courses
4. Lack of Practice/Feedback
5. Graduate Student Teachers
6. Poor Project Management Skills
7. Choice of Language and Objects Early vs. Objects Late

The attrition attributed to factors (1) and (2) can be grouped together as students who are not in the correct program based on their skills or personal preferences. These two factors are not capable of being affected by the system this paper proposes.

Factors (5), (6) and (7) are not addressed in this paper as addressing any of those factors is beyond the capabilities of the system the paper proposes. These factors can be grouped as factors which would require changes to the courses and degrees being offered while the proposed system would work within existing frameworks.

Of those seven factors, two factors, (3) poorly designed CS1 lab courses and (4) lack of practice/feedback are the factors that will be addressed in this paper.

2.5.1 Poorly Designed Introductory (CS1) Lab Courses
According to Beaubouef and Mason (2005), poorly designed labs are a problem that contributes to attrition because students are not getting enough out of the lab sessions. Many universities use the labs in CS1 courses to provide students with time to complete their assignments. At first glance using lab sessions to allow student to complete assignments may seem to benefit students who frequently do not get enough time to program; however, the additional lab time can end up becoming debugging sessions for the instructor and the students (Beaubouef & Mason, 2005, p. 104). Although debugging is an integral part of learning to program, using the lab sessions exclusively for debugging deprives students of the opportunity to learn code design, code documentation, code testing, code reflection and code optimization, among other key aspects of learning to program. Computing, in general, requires multiple skills to be engaged in completing specific tasks, such as the one discussed earlier in the context of programming. This can be substantiated with reference to other introductory CS1 concepts such as computer architecture, operating systems, algorithms & data structures, and the theory of computation.

One of the purposes of the system that will be proposed in this paper is the need to enable students to receive timely feedback about their study skills and study outcomes without requiring the instructor to provide the feedback. This ability to access quality feedback about their code, at least in a semi-automated fashion, allows a re-purposing of the lab time. The repurposed lab time would allow student to spend time developing programming skills, which are not developed through the more common debugging session style labs (Beaubouef & Mason, 2005, p. 104). Furthermore since each student would have individual access to the feedback system, no one student would be able to monopolize the time of the instructor (or the lab assistant) as can happen with the instructor being the only source of aid in the lab.
2.5.2 Lack of practice and feedback

The amount and quality of feedback that students receive is critical as it has a direct bearing on students’ performance. The number of assignments is usually limited, partially due to large class sizes. This coupled with the desire to simplify marking through automation as well as a desire to not incur a substantial time investment for quizzes (Beaubouef & Mason, 2005) means that students are getting a limited amount of feedback. Further, feedback in the form of automated marking is usually not deep enough to provide feedback that is standardized, personalized, and of a high enough quality to meet student needs.

The proposed system in this paper would have the benefit of impacting the two attrition concerns raised under Section 3.5 – poor design of labs and poor feedback mechanism. The design of the proposed system will provide students with quality feedback that is provided in a timely manner. Further, the design also enables instructors to accommodate more assignments or lab exercises for students to work on without necessarily generating a large increase in instructor workload. This could be accomplished by having lab exercises or assignments marked in a formative fashion, for instance, while allowing the system to provide continuous feedback to instructors/lab assistants on student progress. Since providing quality feedback is a time consuming process, being able to offload that work to the proposed system would enable students to get what they need, through extra practice, and instructors to get what they need by not significantly increasing the workload requirements. Such a formative mechanism, among others, also addresses the need for just-in-time feedback for students. It is important to note that formative tracking of competence growth allows instructors and students to customize instructional pathways as study progresses.

3 Feedback

Feedback is defined as “information about reactions to a product, a person’s performance of a task, and so on, which is used as a basis for improvement” (Definition of feedback in Oxford Dictionaries, 2013). In an education setting this is an important part in the student learning process, getting feedback and using that feedback to improve current performance.

3.1 Purpose of Feedback

Feedback is used to achieve multiple purposes in educating students. The obvious reason for feedback is to let students know about their study progress, such as through the grading of assignments to inform them about their mastery of the competencies being tested (University of Victoria Distance Education Services, 2013). Feedback also provides additional benefits such as allowing students to see where their strength and weaknesses are, to keep them on track to achieve the learning of the required competencies for the course, and also to be able to improve their performance in subsequent evaluative scenarios. Another key advantage of feedback is the ability to perform comparative analysis of student performance as well as analyse the quality of instruction.

3.2 Time to Provide Feedback

Providing feedback is only part of the equation. Being able to supply the feedback at the time that is most beneficial to the student is the essence of feedback. Although this will be discussed in more detail
later, one of the key criteria to consider in determining when to provide feedback is that feedback should be delivered when required by students, not necessarily immediately. The usefulness of feedback is reduced if students are not able to incorporate the feedback into the learning that occurs after the assessment upon which the feedback is based (University of Victoria Distance Education Services, 2013). The timely delivery of feedback is also associated with the underlying pedagogy that optimizes the time when feedback is delivered to coincide with the instructional needs of students.

### 3.3 Seven Principles of Good Feedback

Nicol and Mcfarlane-Dick’s paper (2006) provides seven principles of good feedback. This is a widely cited paper when discussing feedback in a post-secondary environment. Many of the seven principles have a direct impact on the design of the system that will be proposed later in this paper. The seven principles of good feedback practice are:

1. Facilitate the development of self-assessment (reflection) in learning
2. Encourage teacher and peer dialogue around learning
3. Help clarify what good performance is (goals, criteria, expected standards)
4. Provide opportunity to close the gap between the current and desired performance
5. Delivers high quality information to students about their learning
6. Encourage positive motivational beliefs and self-esteem
7. Provide information to teachers that can be used to help shape the teaching

While all of the principles are important for ensuring students achieve the desired outcomes from a course, the proposed system will enhance the ability of a course to achieve those principles. The system, at present, is designed to aid students taking introductory courses in programming in a distance education setting. This can be expanded at a later time to address the needs of students taking advanced courses.

#### 3.3.1 Encourage Teacher and Peer Dialog Around Learning

A distance education setting has less opportunity for students to be able to interact with their professors, tutors, and peers because of the potential for higher student to teacher ratios. Also, students are able to complete work at a pace that suits them, which is not necessarily one that facilitates proactive communication. This reduction in communication capability impacts many other principles of instruction. In some distance education settings, communication might even be limited to the returning of graded assignments, which would be a one directional flow of information. The proposed system would add another avenue of communication to introductory programming courses which would help alleviate the lack of communication that is particularly acute in distance education courses.

#### 3.3.1.1 Help Clarify What Good Performance Is

The goals that a student should achieve for a course should be clearly stated. This is important in online courses as students have less of an opportunity to ask questions and get meaningful feedback when wanting to clarify about requirements. If students know what their expectations are, it is easier for them to meet such requirements (Nicol & Macfarlane-Dick, 2006). Expectations are also tied to the regulation
capabilities of students, thus promoting both self-regulation and co-regulation opportunities for students because students will be in a position to compare, on their own, their performances against the course expectations.

### 3.3.2 Deliver High Quality Feedback

A common theme in research into the feedback received by students is that the feedback is not of a high quality (Nicol & Macfarlane-Dick, 2006). The article further defines good quality feedback as feedback which enables the student to learn from their mistakes and move their competence closer to those of the expectations for the course.

Delivering high quality feedback is one of the main goals the design of the proposed project is attempting to achieve. The design, with respect to the domain of coding, aims to be able to provide students with feedback tailored towards their specific areas of need, as determined by the code they are producing to solve assignments and tutorials. The feedback generated will be of a high quality, in a format that best suits the students’ learning needs, which will enable them to be able to more easily close the gap between their current competence levels and those required for the course.

### 3.3.3 Provide Information to Teachers to Help Shape Teaching

Usually, in an academic setting feedback is associated with a flow of communication from the instructor to the student, such as assignment or exam feedback. As noted by the second principle of good feedback, feedback is supposed to include bi-directional communication and instructors can gain valuable insight by receiving feedback from students (Nicol & Macfarlane-Dick, 2006).

The proposed project, by collecting information from students as they complete their programming assignments or tutorials, provides valuable information for being able to more accurately determine what problems students were having when completing the assessment. The problems students were having while solving assignments and tutorials are not necessarily the ones that are left unsolved when the assignments and tutorials are submitted as those will only be ones the students did not have time to solve. Finished assignments and tutorials also do not show which aspects of the completed work were the hardest for the students to be able to solve.

Knowing what constitutes high quality feedback is important to the proposed system as the primary purpose of the system is to provide high quality feedback to students in those areas the students have not yet achieved the desired competency levels.

### 3.4 Types of Feedback

Before taking a detailed look at dashboards, an understanding of the different types of feedback is necessary. This would provide a good base for discussing the type of feedback that will be generated for students, what expectations students have for feedback and the importance of providing feedback in small pieces as well as in concise pieces so that students get the most out of the feedback that is provided by the proposed system.
There are six types of feedback that will be examined and characterised in this section. The types of feedback that will be discussed are formative, summative, concurrent, continuous, affirmative and corrective feedbacks.

3.4.1 Summative Feedback
Summative assessment or evaluation is used to determine how far someone has developed their competencies at a particular point in time (Garrison & Ehringhaus, 2007; Johnson & Jenkins, 2013). Summative assessments are achieved through the use of tests, exams, or other cap stone projects. Summative feedback is when students are asked to provide knowledge gained throughout the term retrospectively from the course in its entirety (Stieger & Burger, 2010). The main use of summative assessment is not to necessarily improve the teaching process as the assessment occurs after the relevant learning has occurred and so cannot be applied (Garrison & Ehringhaus, 2007). Summative assessment should be used to assess how well a program is designed or other higher level measures that do not immediately translate into improvement in instruction and learning. This implies that summative assessment is an assessment type that will have minimal use in the proposed system, as the goal of the system is not to determine if the course works well, but to aid students in achieving the competency goals identified in the curriculum.

The main drawback of summative assessment which makes it marginal for the purpose of this paper is that the feedback generated from summative assessment is not provided to students in a timely manner (Johnson & Jenkins, 2013). This lack of timeliness defeats the purpose of the proposed system as the goal of the system is to provide timely feedback to students to enable them to perform better on course work.

3.4.2 Formative Feedback
Formative feedback or a formative evaluation has the goal of ensuring that the students are learning what they're supposed to be learning. Formative feedback also allows for corrective action in case the desired outcomes are not being achieved by identifying problems and providing students the means to improve in problem areas and to allow instruction to be altered to accommodate the results obtained by formative assessment (Johnson & Jenkins, 2013). Formative assessment is a powerful tool in teaching as it can be employed in a wide variety of educational situations and that the individualized nature of formative assessment is another of its strengths (Sadler, 1998). Garrison and Ehringhus (2007) indicate that formative feedback should be viewed as practice and not have grades applied. This would allow students to look at formative feedback instances as opportunities to improve their competence rather than worrying about the grades that will be assigned.

There are a number of approaches mentioned in Garrison and Ehringhus (2007) and Johnson and Jenkins (2013) that can be used as part of a formative assessment approach, such as observation, criteria and goal setting, and self-assessment. Each of these pieces of formative assessment is important to creating a solid formative assessment platform.

3.4.2.1 Observation
Observation is the ongoing process by which the teachers monitor students' progress in order to determine whether or not they are meeting competence expectations. A key part of observation is that
it requires a detailed level of observation that cannot be achieved by simply determining if the students are doing their work (Garrison & Ehringhaus, 2007). The proposed system is designed to observe, in detail, students on an ongoing basis as the students are completing their work for both tutorials and assignments. With correct analysis being conducted by the system, it will be easy for the teachers to determine if the students are meeting the required competence levels.

The second part of observation is to use the observations to provide students with any additional instruction deemed necessary in order to help meet the competency requirements (Johnson & Jenkins, 2013). The observation portion of formative feedback should be a continuous one, in that students are observed, their observations are analyzed and then a plan for additional instruction, if required, is provided.

**3.4.2.2 Criteria & Goal Setting**

In most cases there is a difference between students' goals in a course and those of the instructor teaching them. Feedback has been shown to be a factor in reducing the difference between a student's goals and those of the instructor (Heron, 2011). One of the purposes of formative assessment would be to reduce or eliminate the gap between what goals a student might have for a course with those goals that the professor and/or instruction. To increase student success in learning course material the academic goals that students are expected to achieve as well as the criteria needed to achieve them needs to be laid out (Garrison & Ehringhaus, 2007). There are various ways in which this can be accomplished, such as rubrics which would create an easily understandable format in which both the skills and knowledge to be learned and the scale to indicate how much progression a student has made towards achieving the required levels of competency in the course.

**3.4.2.3 Self-Assessment**

Another important part of formative assessment is the use of student self-assessment (Garrison & Ehringhaus, 2007; Johnson & Jenkins, 2013; Heron, 2011). The reason for the importance of the self-assessment is that it increases the involvement of students in their learning. Self-learning is more effective when criteria and goal setting of competence levels have been previously set because then the student is able to compare their evaluation of their abilities against the competence criteria established as being required for course competence.

**3.4.3 Concurrent Evaluation**

Concurrent feedback is a form of summative feedback that takes place over the course of a session as opposed to at the end of it. Examples of this are midterms, quizzes, or even a just a few questions. The results of a concurrent evaluation can be used as a supplement for formative feedback by letting the instructor know whether or not a particular concept has been grasped.

**3.4.4 Ipsative Feedback**

In contrast to the previous forms of feedback, ipsative feedback does not directly rely on the results of a particular assessment, but rather compares multiple assessments to determine a learning pattern (Hughes, 2011). In other words it allows students to be able to see whether or not they are improving and developing the required competencies (van Staden & Pilkington, 2012). According to Campos, Mendes, Marcelino, Ferreira and Alves (2012), ipsative feedback is not widely studied but it has the
potential to be helpful in introductory programming courses where students may have problems recognizing improvements in competencies.

Hughes, Okumoto and Wood (2011) are among many papers to note that effective feedback is an integral part of the motivation and learning in distance education. They (2011) note that ipsative feedback can be an integral part of a distance education curriculum as it helps enable the self-assessment mentioned in the previous section by showing students their learning progress. It was further noted that a major drawback of ipsative feedback is that, frequently, the information was not available to allow for ipsative feedback to have the desired effect.

The proposed system would have significant amount of information about students' progress in learning material through the process of data capture while they are working on course assessments. The only limiting factor, especially in a distance education environment, is that the setup of a student's work environment is not within instructor control. It is possible to ensure that students are using a packaged set of tools to record their progress that paves way for the collection of information required to properly provide ipsative feedback.

An article from van Straden & Pilkington (2012) suggests the use of test-driven development (TDD) as a form of ipsative feedback in programming courses specifically in a distance learning environment. The use of TDD gives ongoing feedback about what a student is doing correctly for the assignment and what still needs to be worked on, with the items being done correctly having passing tests while those items needing work being listed as failing their corresponding tests. Although TDD would provide ongoing feedback on how well the assignment criteria is being met, it has a downside where forward progress is not always a guarantee with the use of TDD, as making changes to code might cause previously passed cases to fail. This backwards progress that may result if some tests fail after working towards meeting other requirements does not exactly correspond to the learning of competencies where once a competency is learned moving on and learning a new competency will not make a student lose the knowledge previously gained. This kind of ipsative feedback might be useful for showing students their progress on an assessment, forward progress as well as backward progress, to help students track their competency against course expectations.

3.4.5 Affirmative Feedback
Affirmative feedback defines one of two methods for delivering feedback. This method specifically focuses on providing feedback that tells the recipient that they are going in the right direction. When providing feedback, this method of delivery focuses on the good aspects of the work completed and then highlights which areas need improving.

3.4.6 Corrective Feedback
Corrective feedback focuses on which areas need to be improved as a prerequisite to studying a set of topics to meet required standards. Both corrective and affirmative feedbacks provide the same feedback but the method of delivery varies allowing the feedback to be personalized to the needs of the recipients.
For the purposes of the proposed system, corrective feedback is likely to be easier to implement as the data collected is focused on those parts of the code that the student has done incorrectly. While affirmative feedback might be a better style of feedback to pursue, there would be a significant increase in the analysis of the information generated by the system to be able to determine which areas of competence the student has done well with.

The different types of feedback that can be provided have been examined with the intent of being able to determine which type of feedback, or which combination of types of feedback would be best to use in the proposed system.

3.5 Timeliness of Feedback
Determining the best timeframe for getting feedback to students is an important element in instruction, particularly in online instruction. A common theme in the research on feedback is that the feedback needs to be given in a timely manner. One of the responses that were frequently received by Rowe and Wood (2008) was that late feedback was a common complaint among the students that were surveyed for their thoughts on feedback. Lemley, Sudweeks, Howell, Laws & Sawyer (2007) has classified feedback in terms of how soon after a response appropriate feedback is provided. According to Lemley, Sudweeks, Howell, Laws & Sawyer (2007), the feedback from a graded assignment is the only ongoing communication between an instructor and student that a distance education student will have while taking the course. Many studies have found that the feedback can play an important role in a student's performance (Lemley, Sudweeks, Howell, Laws, & Sawyer, 2007; Rowe & Wood, 2008; Campos, Mendes, Marcelino, Ferreira, & Alves, 2012). These two points would indicate that in distance education more attention needs to be paid to adding instances where just-in-time feedback is able to be delivered to students. The system proposed in this essay would be creating many additional instances where students would be able to receive feedback, as the feedback would be available every time the students compile their code. As the system gets more data from a student, through successive compiles, the feedback that the student will receive would be updated to reflect the current competencies of the student.

3.5.1 The Changing Timeliness
As technology has changed the response time that is expected has grown progressively shorter. Although there is no academic research on the subject, there have been other reports that comment on the demographics of twitter, facebook and other social media users, an article on ragan.com (Bullas, 2012) shows that a large portion of university aged students, more than 25%, have high rates of twitter usage. What makes this an important statistic is that the twitter medium is one of very quick communication, if this is what many students are using in their day to day lives it stands to reason they'll expect more of their communication, including feedback, to be delivered quicker. Other articles talk about the benefits of instant messaging (BigAnt, 2013) over email, the main benefit pointed to is the decrease in response time, specifically that "you can get an immediate response". The proposed system meets this need for more immediate feedback since the feedback it will provide can be given whenever the student compiles data, which means that as soon as the compile gives the student information about any problems in their code they will also have feedback related to those problems.
3.5.2 How Long is Too Long

The question about feedback is when should feedback be provided to students? As noted previously the sooner the feedback is received the better the outcome and learning that can be taken from the feedback. In a traditional learning setting, where the instructor or instructional assistants are responsible for feedback, a major factor that determines the timeliness of feedback is the number of students in the class (Boettcher, 2013). In distance education courses, where typically student to instructor ratio is higher than normal, this would increase the time to feedback when feedback opportunities are already limited from the course being a distance education course. Other discussions state that a hard maximum time between assessment and feedback of 2-3 weeks is the norm (Armstrong, Campbell, Chen, Kershaw, & Milne, 1999). Given the discussion in this section about the increase in expectations for timeliness in the years since that article was written the expected time for receiving feedback has almost certainly fallen. It would then be safe to say that feedback needs to be out to students in no more and likely less time than the 2-3 weeks specified. The proposed system advocates this deadline of getting feedback to students well within a 2-3 week period as it generates feedback on the fly based on the most current data from program compilations that it has received.

Feedback received by students is everything from terrible to amazing and anything in between. A major question that needs to be looked at is how to package feedback, in a timely manner, in order to improve student reception of that feedback.

4 Personality Types and Learning Styles

Everybody has a different personality and each individual learns in a unique fashion. It is important when looking at a system that delivers feedback to examine personality types and learning styles to ensure that the system uniquely meets the learning needs of as many of its users as possible.

4.1 Personality Types

While everyone has a distinct personality, these personalities tend to fall into what are known as personality types. A personality types is a set of personality traits that generalize the personality of an individual. These generalizations, while not perfect reflections of individuals, do allow for analysis of common behaviors that people who have the traits would exhibit. One of the more common and well researched set of personality types are the Myers-Briggs personality types.

4.1.1 Myers-Briggs Type Indicator

The Myers-Briggs type indicator is a way of classifying each person’s personality into one of the sixteen possible combinations of psychological types (Martin, 2011). The 16 personality types results from each person being associated with one of two opposing versions of four different cognitive functions. The classification that a person is found to have through taking a Myers-Briggs Type Indicator (MBTI) assessment is usually considered to be a combination of innate qualities coupled with a person’s preferences for the cognitive functions (Martin, 2011).
The four cognitive functions are - how a person gathers information, how decision making is accomplished, how a person deals with their environment and whether or not a person draws energy from with or from those around them (Kaluzniacky, 2004; Martin, 2011).

4.1.1.1 Introvert-Extrovert
This aspect of Briggs-Myers describes how a person gets their energy. An introvert gets their energy from within, by spending time alone. An extrovert on the other hand gains their energy from interacting with others and working in a group setting.

4.1.1.2 Sensing-Intuition
People with the sensing personality type "prefer information experience and attention to details", while a person who is intuitive prefer abstract concepts and innovative thoughts.

4.1.1.3 Thinking-Feeling
This aspect of personality type is about how a person prefers to make decisions. A person who has the thinking personality uses objectivity in their decision making and is considered to be impartial in their decision making. A feeling person make decision in a more subjective manner and take into account factors that are not strictly logical.

4.1.1.4 Judging-Perceiving
This aspect of personality looks at the structure a person prefers. Someone exhibiting the judging personality tends to be highly orderly and anything they do tends to be well planned. The downside is they may ignore or disregard any facts that do not fit into their established structure. A perceiving person on the other hand do not plan to any substantial degree and are more spontaneous, and they will also adapt better to any facts that do not fit into their existing framework.

The reason why personality types are important is that the personality types a student falls into helps determine how well a student performs in their academic studies, at least among engineering students (Layman, Cornwell, & Williams, 2006). Since many of the traits that engineering students possess are also possessed by computer science students knowing about personality types would help instructors be better able to teach course material.

4.1.2 What Personality Types Suit Computer Science Students
The types of personalities that can be expected to be found in a computing course is important information if the students are to be taught well and learn the material that they're expected to learn. Knowing the personality types that would be more commonly found in a course will aid instructors in being able to tailor the presentation of the material into a format that most suits the students. Despite common misunderstanding about the personalities of students in engineering, there is a reasonable balance between introvert and extrovert personalities in engineering, with introverted being the more common, this contrasts to a marked lopsidedness in the other four measures of the Myers-Briggs Type Indicator which has intuiting, thinking and judging dominating sensing, feeling and perceiving respectively (Layman, Cornwell, & Williams, 2006). Galpin, Sander and Chen (2007) came up with similar findings which showed that for students in computer science thinking and judging were still dominant, but in contrast to the engineering results there is more balance in the sensing/intuition dimension.
Personality types help define the learning styles that are used by students and learning styles, as will be described in the next section, have an impact on how the students want to receive feedback. It is important to know the personality types that can be expected from students in introductory programming courses. Categorizing a student as a particular personality type does not guarantee that student’s affinity for a particular learning style, but it does provide guidelines on which learning styles would be most effective to pursue for appropriate delivery of feedback to that student.

4.2 Learning Styles
Similar to how everyone has a unique personality, each person has a unique learning style. A learning style is a person's preference for the method of learning new information and skills; the preference for learning in a particular way is due to a combination of both inherited and environmental factors (Larkin & Budny, 2005; Alaoutinen & Smolander, 2010). Learning styles are important because it has been shown that when individual learning styles are able to be catered to it improves student motivation and increasing motivation enables students to learn more (Larkin & Budny, 2005). Two prominent ways of classifying learning styles are the Felder-Silverman learning styles and the Kolb model. Both descriptions of learning styles will be described in the following sections.

4.2.1 Felder-Silverman Learning Styles
The next part of the discussion is focused on how the learning style of the students fits together with the different personality types to affect student achievement when taking courses. The Felder-Silverman is a way of classifying learning methods that “focuses on the ways that students absorb and process information” (Zander, et al., 2009). The Felder-Silverman method of describing learning styles is said to be an amalgamation of other methods of classifying learning styles and has a number of dimensions which describe the ways in which people prefer to learn. According to Layman, Cornwell and Williams (2006), there are four dimensions to the Felder-Silverman scale which have the following characteristics, while Zander et al. (2009) include a fifth dimension not mentioned in Layman, Cornwell and Williams (2006). The five dimensions have the following characteristics in how they define a person’s learning style.

4.2.1.1 The Active-Reflective Dimension
This dimension of a learning style is used to determine whether or not someone learns best by experimentation and working together with others (active) or whether a person learns better by thinking things out in solitary contemplation.

4.2.1.2 The Sensing-Intuitive Dimension
The Sensing-Intuitive dimension is described by Layman, Cornwell and Williams (2006) as matching up to the Sensing-Intuitive dimension of the Myers-Briggs Personality Type, which they describe as learning through experience and attention to details compared to abstract concepts and thought processes, where attention to detail is not something that keeps an intuitive person engaged. Zander, et al. (2009) give a slightly different meaning to the dimension giving the difference as being concrete and practical compared to conceptual and innovative. What the two sides seem to come down to is that Sensing learners learn by getting their hands dirty experimenting with a new concept while Intuitive learners learn by thinking about the concept.
4.2.1.3 **Visual-Verbal Dimension**
This dimension is probably one of the most straightforward to understand and are described as being a preference to learning through visual cues such as charts graphs, images, etc. as opposed to learning through written or spoken explanations.

4.2.1.4 **Sequential Global Dimension**
A person who learns on the sequential side of this dimension learns best by being introduced to new material in an orderly fashion, learning one step before moving on to the next step. Those on the global side of the dimension learn in leaps after acquiring all facts.

4.2.1.5 **Inductive-Deductive Dimension**
This dimension is the extra dimension given by Zander et al. (2009); this dimension is used to determine how to present the information to be learned. An inductive learner will learn better if given facts and observation, in other words the concrete aspects, and will use them to learn the underlying principles, while for a deductive learner the opposite holds, they learn better if given the underlying principles from which they will figure out how those principles can applied in the real world.

4.2.1.6 **Computer Science Students Preferences**
Alaoutinen and Smolander (2010) did a survey of students and the results showed a preference for a combination of visual, active and sensing learning styles while both global and sequential learning styles were fairly evenly balanced. In Kinshuk, Liu & Graf (2009) it is noted that students perform better in courses where which supports their learning style preferences. Ensuring that courses are adaptive and can present learning objects to students that compliment their preferred learning style will help those students perform better.

4.2.2 **Kolb Learning Styles**
Kolb's learning style is another framework to identify the learning style. The Kolb learning styles also use a dimension system, which has opposite learning preferences on each dimension. Unlike the Felder-Silverman learning styles which were made up of five different dimensions, the Kolb learning styles consist of two dimensions. The two dimensions of the Kolb's learning style are one dimension related to thinking versus feeling and the second dimension is reflection versus action.

4.2.2.1 **Feeling vs. Thinking**
Feeling or Concrete Experience side of this dimension describes people who are better at sensing what decisions they should be making and tend to be better at relating to other people. The thinking part of this dimension describes people who "think more than they feel" (Larkin & Budny, 2005), these people tend to be more analytical.

4.2.2.2 **Doing vs. Watching**
The doing aspect of this dimension, also known as active experimentation applies to those people who are active participants in situations, while the watching aspect, also known as Reflective Observation, are those who would rather be watching and observing instead of actively experimenting.
The learning style maps people onto a graph where each quadrant corresponds to one combination of Kolb's two dimensions. When a person takes the test their score will place them in one of the four quadrants of Kolb's. Each of the four quadrants has a corresponding learning style.

4.2.2.3 Convergers Quadrant
This learning style is located in the quadrant that combines abstract conceptualization and active experimentation. According to Larkin & Budny (2005) and Gaipan, Sander & Chen (2007), people whose learning preference is in this quadrant like learn through experimentation and tend to do well with labs. They have a 'cut to the chase' attitude and prefer to work alone and work on problems where there is a single solution. Standard lecture formats are also not a learning preference for them.

4.2.2.4 Assimilators Quadrant
This learning style is located in the quadrant that combines reflective observation and abstract conceptualization. This quadrant prefers reflective thinking and is strong in analysis and organization. As with the converger quadrant, this quadrant prefers to work alone, but unlike convergers they do enjoy standard lecture formats (Gaipin, Sanders, & Chen, 2007) (Larkin & Budny, 2005). Since they are focused on ideas and information, they do not learn as well through lab work.

4.2.2.5 Accomodators Quadrant
The accommodators style is located in the quadrant that combines active experimentation with concrete experience. For learning styles people in this quadrant tend to like solving problems and usually approach problem solving with a trial and error approach (Gaipin, Sanders, & Chen, 2007) (Larkin & Budny, 2005). They also enjoy brainstorming and work well in groups being able to communicate ideas and information to other around them, and ask others lots of questions.

4.2.2.6 Diverger Quadrant
The diverger style is located in the quadrant that combines concrete experience with reflective observation. The distinctive aspect of this group of learners is that they have strong preferences to learn in a group setting, particularly through discussions. They are good at coming up with ideas and brainstorming and of all the groups, they have the strongest emotional component in their learning and decision making processes.

4.2.2.7 In Which Quadrant are Computer Science Students?
Galpin, Sander and Chen (2007) conducted a survey of computer science students on learning styles that computer science students prefer. Students in first, second and third year were given the Kolb Learning Style Inventory during the second semester of the year. The results indicate that the majority of students fall into the converger and assimilator quadrants of Kolb learning styles. Just as intriguing was the fact that as students progressed through second and third year this preference for converger and assimilator quadrants was more pronounced. Two possible explanations would account for these facts first. First, those two learning styles are better suited to learning the material covered in computer science. Second, the methods used to teach computer science curriculum are more focused on those learning styles and that if the material could be taught in a fashion that diverger or accommodator learners prefer they might do better. In particular, Galpan, Sanders and Chen (2007) note that the
lecture format that is traditionally used favors abstract reflective learners and note that other teaching strategies that cater to different learning styles should be employed in the classroom.

4.2.3 Personalizing Feedback for Learning Styles
Regardless of which method of determining a student’s preferred learning style is used, students belonging to different learning styles prefer to receive their feedback in different formats. Some learning styles have a preference for feedback to be presented in a visual format, others in an audio format. There are even preferences for the focus of the feedback that a student receives, with some students preferring feedback that focuses on the positive aspects of the work being evaluated before the negative, and others preferring a focus on the negative aspects of the work over the positive. The proposed system, when fully implemented, will be able to provide feedback to students in a variety of formats based on each student’s individual preferences for learning styles. This is a much needed improvement over the typical feedback that students might receive from traditional instruction as instructors have a tendency to provide feedback in a few select formats which correspond to their own preferences for receiving feedback.

5 Proposed Design of an Enhanced ITS
This section of the paper will discuss the proposed design for an enhanced intelligent tutoring system (ITS). The discussion will be started with an introduction into Bloom’s Taxonomy which will provide the system to evaluate the competence of the students. This will be followed with a discussion of the basics of an ITS including discussing the four modules that form the core of an ITS. The section will conclude with a proposal for how the four modules can be implemented and used to enhance traditional methods of learning.

5.1 Feedback and Bloom's Taxonomy
Bloom’s Taxonomy is a system that is used to classify competencies. It employs six levels of classification as a means of determining the complexity of a student’s knowledge related to a particular competency requirement (McMeekin, Konsky, Chang, & Cooper, 2008; Bourque, Buglione, Abran, & April, 2004; Buckley & Exton, 2003; Bruyn & Mostert, 2011).

5.1.1 Bloom’s Levels of Competency
The six classifications in Bloom’s Taxonomy are Knowledge, Comprehension/Insight, Application, Analysis, Synthesis and Evaluation, and each of these will be discussed in a bit of detail.

5.1.1.1 Recall
This is the most basic level of competency. In order to achieve this level, a student would be required to have recall or memorization of specific facts pertaining to the competency being learned. In programming this could be being able to correctly list the possible layouts of a conditional block (i.e. if . . . else if . . . else) (McMeekin, Konsky, Chang, & Cooper, 2008). To achieve this level of competency understanding of the information that has been memorized is not required (Buckley & Exton, 2003).
5.1.1.2 Comprehension
The second of Bloom's levels of competency, comprehension focuses on being able to understand the problem domain (Bruyn & Mostert, 2011). This would be expressed in learning to program by being able to clearly state what a particular competency does, such as accurately determining the outcome of running a given piece of code using a sample input.

5.1.1.3 Application
Application is the third level of Bloom's Taxonomy and achieving the application level of competency requires students to apply competencies that have been learned to the comprehension level to new problems (McMeekin, Konsky, Chang, & Cooper, 2008). Achieving this level of competency would require that students be able to determine for themselves which programming structure would need to be used to solve a given new problem (Buckley & Exton, 2003). An example of this would be for a student to develop a method given an abstract specification that does not include specifics on appropriate data structures or algorithms.

5.1.1.4 Analysis
The analysis level of Bloom's Taxonomy focuses on being able to break down a problem or domain into its component parts (Buckley & Exton, 2003). In order to demonstrate this level of competence a student might be required to explain the role a particular piece of code or a particular function/method plays within a given system (McMeekin, Konsky, Chang, & Cooper, 2008). Another way of demonstrating this level of competence is to assess a student's ability to create diagrams that indicate relationships between the various components of the system or the flow of control through the system.

5.1.1.5 Synthesis
Synthesis describes being able to take the various concepts of the competency being learned and apply them to a new problem or a domain that provides a new meaning or structure. Synthesis level knowledge is described by Buckley & Exton (2003) as building a new "whole" from the different pieces of knowledge learned in lower levels of the taxonomy. McMeekin et al (2008) provide an example of synthesis in programming of creating new functionality, as opposed to the Application level where existing functionality is altered as opposed to created. At this level of competence a student would be expected to able to transfer knowledge from existing pieces of knowledge to new and unique pieces of knowledge. For instance, a student could observe feedback analysis in a traditional classroom setting and encoding/enhancing it as part of an automated feedback mechanism in an online learning setting.

5.1.1.6 Evaluation
At this level of competence a student is able to make judgments on the merits of a proposal in the target domain (Bruyn & Mostert, 2011). The judgment would be well supported to demonstrate a deep understanding of the material. In a programming environment this might take the form of an appraisal of a particular software solution. In general such a level of competence would be beyond the scope of an introductory level programming course. It is interesting to note that Bourque, Buglione, Alban & April (2004) equate this level of competence with being a domain expert. Clearly, this would be a level that is not expected to be achieved in an introductory programming course and that those at this level of expertise would be expected to be able to write a research paper on a subject in the domain. It involves
the ability to assess whether a particular judgement is supported by observed data. For instance, in the example noted under ‘Synthesis’, the student could run an experiment to determine whether the automated feedback mechanism indeed mirrors what was observed in the traditional classroom setting.

5.1.2 Bloom’s Taxonomy & Feedback
Bloom’s taxonomy can be utilised to determine feedback mechanisms. In general, the further along students get in Bloom’s Taxonomy the less detailed feedback they should be receiving. How far along in the taxonomy a student is should be determined by whomever is providing the feedback. In the proposed project, the system is expected to collect this information either directly by asking the student himself/herself, or indirectly using analytics methods. If a student has been determined to have reached the level of competence required for a skill then the level of feedback should change its focus on that skill even if additional mistakes are observed related to that skill. For example, if a student is deemed to be proficient enough with loops to meet course competency requirements, then if the student makes a mistake with loops at a later point, then very little or no feedback should be provided or optional feedback that is distinctly separate from feedback for areas where the student has not yet achieved the desired level of competence. The feedback for already achieved competence could be detailed in a manner such that the student is getting a refresher on a topic they have achieved competence for, as opposed to feedback detailing areas that require improvement. In terms of programming, this type of distinction is important as even the most seasoned programmers will occasionally make mistakes in those areas of competence in which they are well versed and should not be given feedback that would be the same as for someone who is just developing their skills in that area of competence.

For the proposed system, the question to be answered is whether or not the compile data will be sufficient to enable the system to determine what competency level of the taxonomy the student has achieved. Even with relatively clear descriptions of the different levels of competence the wide variety of possible inputs, such as similar code with different variable or method names might make such an analysis difficult though not impossible.

5.2 Intelligent Tutoring Systems
An intelligent tutoring system (ITS) is a system that teaches a subject interactively and through the use of learning tools. An ITS will provide individualized feedback and learning paths through the use of artificial intelligence techniques. A specific advantage of an ITS, especially in the context of distance courses, is that a teacher is not required for an ITS to be used. An ITS can be operated either on its own or in conjunction with an actual human teaching resource.

5.2.1 Background
Starting as early as the 1950’s computational techniques were being used to aid in teaching and as computers and the programming languages that were available to create programs got more complex so did the programs used for educational purposes (Ke & Lu, 2010). This lead to the development of computer aided instruction (CAI) whose goal was to create a series of detailed lessons on a topic tailored for each student (Rahbari, Meech, & de Silva, 1997). Many of these early CAIs were “simple electronic page-turners” (Rahbari, Meech, & de Silva, 1997) and had little or no artificial intelligence components. With the incorporation of artificial intelligence into CAI the systems started becoming more advanced...
and are able to analyze and change the behavior of the CAI based on the interactions of the learner with
the CAI (Rahbari, Meech, & de Silva, 1997). A CAI with artificial intelligence components to aid in
instructions are known as intelligent tutoring system. At the time of writing Rahbari, Meech and de Silva
(1997) CAls intelligent or otherwise were used as a supplement to traditional classroom teaching. Due to
research over the last 15 years ITS are now being readied to be used in larger roles in the teaching
experience particularly in distance education or other environments where student to teacher
interaction might not be as feasible. What separates an ITS from CAI is the ITS' ability to diagnose and
adapt to the changing situation and stimulus provided by the student (Yamna, Mellouli, & Wuillemin,
2010), such as the answers provided to questions, or as in the proposed project by the reports that are
created when building a project. The adaptability of an ITS is further extended through the interactions
with the student changing based on their learning style of choice (Rahbari, Meech, & de Silva, 1997). The
first part of the adaptability of ITS, changing the concepts taught based on student interaction can be
considered to be deciding what to teach students, while the second part of adaptability, catering to the
various learning styles of students, will answer the important question of how.

5.2.2 Structure of Intelligent Tutoring Systems
There are four key parts to the structure of an intelligent tutoring system. The four parts, each of which
will be discussed in a bit more detail below, are domain expertise, student model, teaching expertise
and interface. None of the four parts of the ITS can be neglected being deficient in any one of them
hurts the quality of the system as a whole.

5.2.2.1 Expert Module
The expert module is the part of the intelligent tutoring system that contains all the relevant knowledge
for the subject area that the intelligent tutoring system will be used to teach (Ramesh & Rao, 2012). The
module can be known by various names such as the domain module (Srisethanil & Paker, 1995). The
expert module contains information that can be separated into two main categories, the high level
category and the low level category. The difference in information between these two categories is that
the high level information is information that should be known before starting to use the ITS, while the
low level category are the facts and information that are expected to be learned through using the ITS.

5.2.2.2 Student Model
The student model can consist of various bits of data that give the system the information it needs to
 tailor output to what a student has previously accomplished and in the learning style that the student
prefers (SunandanChaakraborty, Roy, Ghomnick, & Basu, 2010). At the minimum the student model,
also referred to in literature as the user model, contains the data that specifies what knowledge the
student currently possesses in order to be able to provide appropriate lessons and feedback.

5.2.2.3 Teaching Model
The teaching model, also referred to as the tutoring model or control engine, is the part of the ITS that
takes the input that the student provides and returns customized output based on the students current
competencies and their preferred learning style (Yamna, Mellouli, & Wuillemin, 2010). More importantly
from the point of view of the system that will be proposed later in this paper, the teaching model is also
responsible for using the student input to determine any gaps in student knowledge and then to provide
remedial tutoring in a format that is compatible with the learning style preferences expressed in the student model (Yamna, Mellouli, & Wuillemin, 2010; Rahbari, Meech, & de Silva, 1997; Srisethanil & Paker, 1995; SunandanChaakraborty, Roy, Ghowmick, & Basu, 2010)

5.2.2.4 Interface Module

The interface module is also called the communication module (Rahbari, Meech, & de Silva, 1997) or the listening agent (Ke & Lu, 2010). The interface of the ITS is of key importance to how well the intelligent tutoring system works and its quality is a determining factor in the success of an ITS (Rahbari, Meech, & de Silva, 1997). The interface module controls the information that passes back and forth between the student/user of an ITS and the implementations of the other models (Rahbari, Meech, & de Silva, 1997).

In this particular setting, since the intelligent tutoring system is being used as a replacement, partial or complete, for instructor/student communication, there would actually be two separate interface modules in the proposed system. The first interface module, as with a standard intelligent tutoring system, would be for the students to interact with the system. The second interface module is for the instructors to interface with the system. The interface that the instructors will use would display data in a dashboard style showing information such as statistics on what type of errors were the most common and other data that would help instructors teach further material or enable instructors to focus more on areas of competency that students are struggling with.

5.2.3 Intelligent Tutoring Systems and Learning Styles

In order for an intelligent tutoring system to be able accommodate different learning styles that students' posses each of the four components of an intelligent tutoring system must be able to accommodate information in different learning styles. The domain expertise component must have its expertise recorded in a fashion that allow for the same expertise to be retrieved from multiple different starting points. The student model must be able to recognize the different instructional preferences a student might have, usually these are initially gathered when the student first uses the system by providing a survey so that students can identify their starting learning style preferences. The student model must also be able to analyze how the student actually performs based on completing various tasks set by the ITS, feedback from this analysis will determine if the student's preferred learning style has changed and to cause an alteration, if required, in how the material is presented and in the exercises that the student is asked to complete.

A standard ITS will provide its own assessments to determine the current level of student competency. The proposed system will not be providing any assessments on its own but will instead rely on the compile data submitted by students while completing their assignments, labs and tutorials. This reliance on outside data will make the teaching module more difficult to create as the input to be analyzed will be more varied than the data generated through a more fixed set of assessments.

5.2.4 Intelligent Tutoring Systems and Learning Analytics

Analytics is the analysis of large data sets to convert that data into information for the purpose of making decisions or evaluations. Learning analytics is the analysis of large data sets generated by students interacting with their learning environment (Siemens, 2012; Morris, 2011). We can extend this definition to data sets that are generated by the entire learning context of an individual student or
groups of students. The entire learning context includes data sets generated by students, data sets generated by instructors, data sets generated by student’s peers, data sets generated by instructional systems associated with the learning context, and any other related data sets.

The analysis and discovery of relations between human learning and contextual factors that influence these relations have been one of the contemporary and critical global challenges facing online learning researchers. Traditionally, these relations concern learner performance and the effectiveness of the learning context from a summative point of view. Tracking marks distribution in assignments (say, learning efficiency) and the instruction targeting topics covered by these assignments (say, instructional effectiveness) would yield a deeper understanding of the evolution of learner competency at the personal level, while expanding the scope for quality feedback.

Learning efficiency encompasses any and all aspects that concern “learning” of individual learners or groups of learners. Examples of learning efficiency aspects include learning style, metacognitive scaffolds, peer interactions, self-regulation, co-regulation, social networking, and other learning-oriented activities and characteristics associated with learners.

Instructional effectiveness encompasses any and all aspects that concern enhancement of targeted as well as inadvertent “support for learning”. Examples of instructional effectiveness include pedagogy, andragogy, peer evaluation, software-agent-oriented guidance, lectures, content, presentation of content, instructional design, learning objects and other resources, assessment structures, open learning, and so on.

Contemporary studies that relate learning efficiency with instructional effectiveness were situated in a summative approach where the interplay between instructional intervention and learner performance is clearly demarked by a well-defined study activity as in a final exam or a midterm exam or a project.

With the advent of new technologies such as eye-tracking, activities monitoring, video analysis, content analysis, sentiment analysis and interaction analysis, one could potentially engage in repeating portions of experimentation performed under the summative approach but with the inclusion of formative data. This would not only enable one to understand the relationship between process-centric formative data and the product-centric summative data, but would also allow one to address learning efficiency and instructional effectiveness at an optimal granular scale. One can vary the scope of contexts before determining the optimal size of scope that is beneficial to both the student and the instructional resource. This very notion is what is currently being explored under the aegis of big data learning analytics, which includes sub areas such as learning process analytics, institutional effectiveness, and academic analytics.

Big data analytics, as opposed to smaller data analytics that can be equated to approaches that use data mining or simpler artificial intelligence in education techniques, targets large volumes of data (beyond terabytes) as well as large numbers of voluminous computational models (models using a significant number of variables) concerning learning efficiency and instructional effectiveness. Big data learning analytics is all about analysis of learning patterns in various yet related levels of granularity from the large volumes of data and the large numbers of associated models. For instance, sentiment analysis can
be performed to computationally capture relationships between learner mood swings and competency growth among students registered in a program.

Big data is characterised using the following five factors – volume, speed, variety, veracity, and value. Volume addresses the sheer quantity of data that is expected to be in orders of magnitude much higher than Gigabytes. Speed refers to the arrival and processing of learning related data; while it is discrete in nature, it could be sufficiently dense to be treated as a continuum. Variety implies that learning related data could be structured, semi-structured, unstructured, interconnected, and discoverable. Veracity advocates the need for information extraction, model development, and machine-oriented learning to adhere to quantifiable truth in BIG data, which requires multiple levels of validation of the underlying data and its transformation. Finally, Value dictates that the utility of recognizing, discovering, and promoting learning related patterns, aka learning traces, be explicitly associated with performances of individuals, instructors, institutions, and software agents.

Learning Analytics is different from Intelligent Tutoring Systems in terms of the specific focus on learning evolution and in terms of data approaching BIG data. Learning Analytics is also different from Educational Data Mining in that it does not expect well-defined data to be available in a repository.

Learning Analytics has an important implication for the proposed system as the system will be doing analysis on large amounts of student data generated by the students compiling their code while working on their assignments. If students are compiling at reasonable intervals while they are completing their assignments, labs and tutorials, then the amount of data generated will quickly grow beyond other methods of analysis. Projects are underway in Athabasca University to explore the application and effectiveness of learning analytics in online learning scenarios.

6 Conclusion
Having discussed and characterised student attrition with respect to feedback types, personality types, learning styles, cognitive knowledge levels and technologies (ITS), this section will attempt to synthesize factors discussed so far in the form of a system proposal before highlighting some of the challenges and future work associated with the proposed approach to attrition reduction.

6.1 Summary
The system is designed to target novice programmers in higher education and activities related to learning to program. When students are programming an assignment or tutorial, they usually follow the contemporary programming cycle of writing code, compiling the code, debugging the code, and testing the code before submitting. Opportunity for feedback is limited mostly to summative feedback after the code has been submitted for evaluation. If students code in a lab environment, there is the possibility of formative feedback from lab assistants. In an online environment, the formative feedback is restricted mostly to asynchronous communication between the student and his/her tutor. Some intelligent tutoring systems offer context-specific feedback but such feedbacks are highly limited to a specific set of well-defined problems. Feedback oriented around personality types, learning styles, and cognitive knowledge levels are not available in contemporary coding environments.
The proposed system enhances the contemporary model with a view to a) track each student’s coding habits with respect to writing of code, debugging, testing, documenting, optimising, regulation of coding habits, b) record coding habits of students in ontology, c) analysing coding habits with respect to personality types, learning styles, and cognitive levels, developing computational models of coding habits, offer just-in-time, context-specific, personalized, and paced-synchronisation feedback, d) improving traditional ITS that the proposed system replaces, and finally e) assess the impact of custom changes in the system as a measure of instructional effectiveness and learner competency.

a) tracking each student’s coding habits

A variety of integrated development environments (IDE) exist for students to code. Eclipse and NetBeans are two popular IDE that students use to learn programming. Being open systems, these IDEs allow researchers to extend their functionality to facilitate tracking of each student’s coding habits. Presently, we have extended Eclipse to capture key events such as compile errors and warnings, creation and modification to classes and methods, documentation efforts, and testing efforts by students. With compile events capture, we record the code that has been compiled under each compile event, errors and warnings generated under each compile event, and the time at which the compile event happened.

When completing the compile step of the programming cycle the system will record as much data as possible about the current state of the assignment. The state will include such information as the current breakdown of the code structure, including the classes, methods, functions, the parameters of the various functions and methods and any other details that can be extracted. The state should also include any errors or warnings that were generated at compile time. Ideally the system would also be able to monitor the programs while they are running with the goal of capturing any compile time exceptions, with the corresponding stack trace.

The data also includes code design activities, code documentation activities, code debugging activities, code testing activities, code reflection activities and code optimization activities.

b) recoding coding habits in ontology

Raw data observed from the IDEs can be translated into ontologies. For instance, one can extend a Java syntax ontology with errors/warnings, debugging, documentation, tests, optimization, and reflection. The resultant ontology could be visualized as a table depicted below.
Each entry in the ontology is time-stamped, hence allowing researchers to observe occurrences of specific skills or gaps in knowledge over a period of time. Such time-wise traces will enable estimates of competencies observed in each student. For example, if a student is found to frequently incur errors and warnings corresponding to control structures for the first 4 weeks of an introductory programming course, and is found to sparsely incur such errors and warnings for the remaining 8 weeks of the course, then one can surmise that the student had made significant strides in acquiring coding competency with respect to a specific set of control structures in Java.

Further, the ontology can also relate debugging efforts of a student with respect to specific errors and warnings by tracing the amount of time taken by the student in resolving a particular type of error. Going one step further, by tracking the process followed by the student to debug this error, the system can track the quality of the student’s debugging process.

Similarly, other dimensions of students’ coding habits can be traced and ontologized into clusters of competencies and competency gaps.

A sample Java language ontology is presented in Appendix 1.

c) analysing coding habits

The coding habits of students thus clustered in the ontology can be further categorized with respect to various personality types, learning styles, and feedback types. Some of this information could be obtained by asking the student explicitly or by inferring the information implicitly. Evidences of compatibility between the triplet – 1) student’s observed behaviour in coding, 2) student’s observed level of performance (preferably using Bloom’s cognitive levels) and using results of other assessments, and 3) student’s observed personality/learning-style/feedback types can be estimated using information stored in the ontology. Feedback could be offered across multiple media, using various communication channels, and by offering mixed-initiative dialogs (system-to-student initiative or student-to-system initiative, for example). If the student prefers, the system could restrict feedback to one-way system initiated feedback.
Similarly, results of any change in learning styles or feedback types can be attributed to observed performances and/or observed coding behaviour.

Student’s performance can also be measured with the introduction of self-regulation and co-regulation activities. Research indicates that regulation improves performances. Such a contention can be directly tested using the proposed system.

d) improvements over traditional ITS

The proposed system will provide four improvements over a traditional ITS. The first improvement is that the students will be working in an IDE as opposed to working an environment that is integrated into traditional ITS. This improves traditional ITS by allowing students to work in an environment that will be useful to them even after they have completed working on the ITS, this also frees up development time from creating a built in environment to other aspects of the ITS, such as determining where students lack competency and providing personalized feedback to help them achieve those competencies. The second improvement over a traditional ITS is that the proposed system is flexible to changes in pedagogy as well as instructional design as the assignments can be changed or reordered without any effect or major change required to the system. The third improvement is that the system will be providing personalized feedback in a form determined by the learning preferences of the individual student, this compares to traditional ITS having very few different methods of learning. The fourth improvement is the amount of data collected and analyzed for an individual student and overall. In a standard ITS there is limited data to be collected and used to due to the manner in which ITS’ determine student competency. The proposed system, due to using student compile time data, will have significantly more data points for use in analysis to determine student competencies which, with proper analysis, will provide better and more accurate assessments of a student’s competence level as well as enable tracking of changes in competence levels over time.

e) assessing the overall approach

Presently, a set of studies are being conducted at the NSERC/iCORE Laboratory under the guidance of Prof Vive Kumar, Prof Sabine Graf, and Prof Kinshuk to characterize and assess the impact of customized feedback and customized learning styles on learner performances. One such study explores the impact of learning styles on learner achievement. Another study explores the impact of learner mood swings on learner satisfaction. Yet another study explores the impact of learning analytics on learner performances. All these studies point to the need for bigdata learning analytics as the basic framework for data collection, clustering, mining, and feedback. There are significant prospects for further research with the proposed system. The research can be focused in a number of areas; particularly to answer the question - does the system provide actual benefits to students? From an instructional point of view, instructors should start making use of the data in order help determine those course materials that students are having difficulty comprehending. This is particularly important as this provides an
additional avenue of feedback that is both timely and significant. Such feedback can be given in class soon after the data is received by the instructors allowing instructors to focus on areas of difficulty thereby improving the quality of their instruction.

6.2 Future Work
The implementation of the proposed system is not an end point in a journey but rather a starting point. This paper has demonstrated that attrition and dropout rates in computer science and other related disciplines is a problem, and the problem is worse for certain groups, especially in distance education settings. Of the problems that students have said contributed to their decisions to drop an introductory programming course, this paper tackles the issue of providing quality feedback to students in a timelier manner. A system design has been presented that caters to this very need. While there are many conceptual challenges to the system that associates scope, scalability, appropriateness, and level of customization, this paper contends that such a system would be essential to successfully reduce attrition and drop outs.
7 References


University of Victoria Distance Education Services. (2013, 06 04). Providing Feedback on Assignments and Activities. Retrieved from University of Victoria - Continuing Studies Distance Education Services: http://distance.uvic.ca/pdfs/teams/Providing-Feedback-on-Assignments.pdf


Appendix 1 – Java Ontology

Java
- JavaLanguage
  - Data
    - Expression
      - ActualMethodParameter
      - ArithmeticExpression
      - AssignmentExpression
      - BitShiftExpression
      - BitwiseExpression
      - ClassMemberInvocation
      - ConditionalExpression
      - IncrementDecrementExpression
      - LogicalExpression
      - Operator
      - TypeCasting
    - Identifier
    - ProgramStructure
  - Statement
    - DefinitionStatement
    - ExceptionHandlingStatement
    - ExpressionStatement
    - ImportStatement
    - IterationStatement
    - JumpStatement
    - MethodInvocationStatement
    - NestedStatement
    - PackageSpecificationStatement
    - SelectionStatement
    - StatementBlock

JavaOOP
- Class
- Constructor
- Interface
- Method
  - AbstractMethod
  - FinalMethod
  - JavaStandardLibraryMethod
  - StaticMethod
- Object
- OOPPrinciple
  - Abstraction
  - Encapsulation
  - Inheritance
  - Interfacing
  - Polymorphism