International Journal of Education and Development using Information and Communication Technology (IJEDICT), 2009, Vol. 5, Issue 3, pp. 85-100

Impact of using agile practice for student software projects in computer science education

G.I.U.S. Perera
University of Moratuwa, Sri Lanka

ABSTRACT

Computer Science education is becoming a fundamental teaching area with the Information and Communication Technology (ICT) development. It is a known fact that traditional educational and teaching methods have certain limitations for ever changing technology based and software & tools interactive courses such as teaching programming or software development. After years of been practiced in the industry the Agile software development process possesses standard characteristics of a process paradigm. However, it is rare to observe studies on Agile practice used in computer science education with its impact to student learning. This paper describes findings of such study conducted in a university education environment. The study was conducted on a student programming project, with sample size of 100. The results indicate a significant impact on students' skill improvements up to 29.23% at best case. It also shows, a reliable method of improving relatively weaker students' programming skills, showing fascinating average skill variance between their project mates, reduction from 0.508 to 0.209. Furthermore, Agile process practiced students have shown 6.33% and 5.65% higher marks for coding and final evaluations of their projects on average, over the controlled experiment sample.

Keywords: *Agile process, Pair Programming, Computer Science Education, Teaching Programming, Student Software Projects, Learning Development, ICT Learning*

INTRODUCTION

The fruitful education is a fundamental necessity for human development. Throughout the human civilization there have been many distinguished learning and teaching methods developed and used. With the technological advancements in different disciplines, educational methodologies and norms have to be evolved from time to time. Due to the increasingly diverse population, education is changing toward a more global, technology-rich environment designed to meet these diverse and changing needs of students (Gunter, 2007). One of the most valued benefits that Information and Communications Technology (ICT) affords over traditional teaching practices is its capacity to extend the student's learning beyond the actual limitations of the classroom. This enhancement of the student's learning refers not only to the place, but also to the time and people that are involved in the process (Arbelaiz & Gorospe, 2009).

> *"Also Rapid advances in ICTs demand changes to our education systems … While most educators appear to acknowledge the importance and relevance of Information and Communication Technologies within teaching, difficulties nevertheless continue to be experienced within the processes of adopting these technologies" (Knight et al., 2006).*

Beyond that teaching computer programming, which is indeed in a well ICT enabled learning environments needs more technological enhancements to the teaching activities to develop further. The potential value of effective support tools and methods helping students grasp difficult programming concepts is even greater (Domingue & Mulholland, 1997). This emphasises, that teachers for computer science need to work further steps ahead of just using ICT enabled

environments to teach the students; one area may be to incorporate cutting edge ICT tools, practices, and processes with the learning activities.

As much like with other disciplines, teaching computing and ICT industry is highly interdependent. Student projects are done with the collaboration of industry and industry expects rapid absorption of fresh talents to their mainstream projects without any advance training efforts. The Department of Computer Science and Engineering (CSE) at University of Moratuwa, maintains a good rapport with the Sri Lankan ICT industry making mutual benefits to each other. There have been frequent curriculum revisions to accommodate industry needs while preserving the merits of university education. This research is the effort to examine the suitability of using heavily industry utilized software development process practice in student software projects.

The organization of the paper is as follows. The next section discusses the background literature of the research in a brief manner covering the areas of computer science education, Agile software process, and pair programming. After that the research problem, which initiated the necessity of novel approaches for teaching programming is described. The experiment methodology explains the process carried out for the research including the experiment background and experiment setup with key parameters used. Thereafter, a comprehensive analysis based on the experiment results is included. The analysis section rationalize the argument of this paper that Agile process shows significant positive results on student programming projects supporting them to grasp complex skills more easier. The analysis also includes a questionnaire data analysis on experiment sample students' feedback of the process they followed. The paper also discusses the experiment limitations and their relative impact to the study observations to facilitate fair and neutral judgement for the readers. The conclusion draws readers' attention towards possible further researches and policy implications to be drawn, while summarizing the research in brief. Finally, the references complete the paper.

BACKGROUND LITERATURE

This section includes a brief summary on literature which was mainly used as the basis for this study. One could find a reasonable amount of literature on education methods and development approaches relevant to the university education, including ICT education. However, for this particular study, a specific focus computer science education and Agile methods was given as they provide the stem of this research. Importantly, the Agile software development methods have proven its success in the industrial projects, but not much examination were done on student projects context. Therefore, even though this study entirely used the Agile process in student projects context, the literature might appear more towards industry based findings. Nevertheless, there is no difference on using Agile practices in either context, hence the findings from literature can be considered as valid sources.

Computer Science Education

Teaching computer science has never been a straightforward or simple process. As a result, a great deal effort has been aimed at improving the teaching process (J. Domingue, P. Mulholland 1997). There have been a number of studies, which have attempted to identify Information Systems (IS) graduate skills and the resultant curriculum (McCarthy and Hawking, 2002). Some researchers have tried to define a new research area for computer science education taking it in isolation to the traditional education system. This was heavily criticized by Almstrum and others as

> *"The real challenge in computer education is to avoid the temptation to re-invent the wheel. Computers are a revolutionary human invention, so we might think that teaching and learning about computers requires a new kind of education. That's completely false:*

The basic mechanism of human learning hasn't changed in the last 50 years" (Almstrum et al., 2005).

Furthermore, Mason and Weller indicated that …the conditions which are needed to produce good educational discussions are far more complex, more people-dependent and more educationally determined than mere technology will ever influence very significantly (Mason and Weller, 2000). However, as Covington expressed, lack of clarity about the benefits of technology, lack of willingness to take risks, and the need for more rigorous course planning have deterred some academics 'entrenched in traditional tools and pedagogies' from changing familiar instructional practices (Covington *et al*., 2005), is one of the main reasons which explains why academics do not much research on complex software technologies to improve computer science education. Furthermore, the emphasis on technology in education is not to imply that the technology is the goal of the educational process; however, a technological learning environment can alter the way students learn and the way professors teach (Culp *et al*., 2005).

"Access to new technologies, the changing nature of higher education and an increasingly diverse student population highlight the need to review the ways higher education is delivered" (Birch & Sankey, 2008). This emphasise the necessity of doing continuous research on possible improvements for the teaching methods. One of the most difficult concepts to teach is adopting a critical position in relation to inquiry about digital technologies (Arntzen, *et al*., 2008) is a good statement with respect to identifying possible new approaches for overcoming issues in teaching ICT disciplines. Computer programming still remains an important part of most Information Systems courses. However the emphasis today is on teaching programming concepts and style and using programming languages to support this objective (McCarthy and Hawking, 2002).

Agile Software Process

Agile software process was defined as collective nature by group of experts to overcome issues with the traditional software processes. Agile Manifesto was the proper introduction of the agile methods to the software industry. According to the Agile Manifesto the following four norms are the basics of the Agile methods.

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan (Agile Manifesto, 2001).

  "Agile Methods are a reaction to traditional ways of developing software and acknowledge the need for an alternative to documentation driven, heavyweight software development processes" (Cohen et al., 2003).

In most of the traditional software processes, there are heaps of documents when the project finishes. Despite from those most obvious differences between plan-driven life-cycle models and agile development is that agile models are fewer documents oriented and place more emphasis on code development (Perera & Fernando, 2007). By the nature of this paradigm it also provides some other benefits like, flexible project management, cost effective adaptability (Perera & Fernando, 2009), increase communication and ultimately increased customer satisfaction (Perera & Fernando, 2007).

There are many principles behind the agile practice. Some of them are based on behavioural and managerial improvements to the software development (Agile Manifesto, 2001). The development process is flexible and agile practitioners believe for different projects, different approaches and process models have to be used. Agile process welcomes frequent requirement changes even at late stages of the project. With frequent deliverables, agile process measures its progress through the norm of working software (Dagnino, 2002). The Agile philosophy promotes new ideas of system development that contrasts traditional methods (Highsmith & Cockburn, 2001).

Importantly, Agile type of thinking has initiated a massive paradigm shift in the software development arena. Even Agile does not cover the entire spectrum; this attitude change was the significant achievement.

A simple but comprehensive definition about identifying an Agile project by Abrahamsson and others in their book explained as

> *"What makes a development method an agile one? This is the case when software development is incremental (small software releases, with rapid cycles), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify, well documented), and adaptive (able to make last moment changes) (Abrahamsson et al., 2002).*

Pair Programming

The idea behind pair programming, also known as Collaborative Programming, is straight-forward. It involves two programmers collaborating side-by-side on the design, coding and testing of a piece of software. One, the Driver, controls the keyboard/mouse and actively implements the program. The other, the Navigator/Observer, continuously observes the work with a view to identifying tactical defects and providing strategic planning (Lui & Chan, 2006). The pair programming is one of the core practices in the Agile process.

Williams *et al*. (2003) have done extensive research for using pair programming as learning method and their work extended to university programming courses. Students who performed pair programming was able to produce more code and their code were easily readable (Bipp *et al*., 2008). Students who followed pair programming shows competence in solo programming (individual programming) activities and possess positive attitudes towards programming courses (Williams *et al*., 2002). Another important characteristic with the pair programming is the peer evaluation of each others programming activities. This is similar to having a mentor next to the student throughout his work which is really impossible to achieve in university education. Peer assessment is known to have positive effects on student satisfaction and learning effectiveness in different disciplines in higher education (Gatfield, 1999). There is evidence that encouraging novice programmers to self-explain and critique their impasses improves understanding (Davis, Linn, & Clancy, 1995), This may be difficult with a superior person such as teacher, assessor but can be easily done with own colleagues, peers.

RESEARCH PROBLEM

There were several issues with the present learning process of computer programming, laid the step stone for this research. Out of those the lack of collaborative learning environment for students was significant. "From a pedagogical and constructivist perspective, collaborative learning methods tend to encourage construction of knowledge, deeper understanding and greater skill development by their ability to engage students dynamically in the learning process" (Alavi, 1994). In fact the Agile process has shown its ability to form a collaborative software development environment in the industrial practice (Perera & Fernando, 2006).

According to Kember's description, there are two kinds of broad orientations in teaching: the teacher-oriented conception and the student-oriented conception (Kember, 1997). The teacher-centred learning strategies are described as focusing on the teacher transmitting knowledge from the expert to the novice. In contrast, the student oriented learning strategies are to focus on the students' learning and 'what students do to achieve this, rather than what the teacher does' (Harden and Crosby, 2000). This is what exactly the students require in their programming courses. However, on the other hand there is danger of students' frustration if they allowed their

own learning, especially when it comes to bug fixing and compiling the code without errors. This indicates a new paradigm of learning with significant autonomy for students is required for improving.

Another problem area with student programming projects is that even though the students were allowed to practice group working the evaluation criteria are mainly based on considering the outcome. In such scenario how the group has able to produce the results are not much relevant (Johnston & Miles, 2004) What actually happens in many cases is the students work their parts individually and try to integrate them together at the end making the peer-to-peer interactions minimum (Vik, 2001) and not achieving the expected goals of group working. This creates another major issue with weak students in the group projects where they struggle with their parts to complete where as the competent students have already completed. At the end knowledge improvement of weak students will not happen as expected. Furthermore, their inability to meet with group competent level make them as a burden to the group and their programming life becomes a misery for them. This could easily make students to cheat and plagiarize others' works which teachers do not tolerate on any circumstances.

Team work and the clear presentation of ideas more accurately reflects the current tasks of the industrial programmer than sitting alone carrying out a programming assignment (Dawson *et al.*, 1992). Effective teamwork requires mastering specific abilities, such as leadership, coordination and conflict management. This implies that if higher education wants to meet the requirements of the students' future professional lives, it has to address the acquisition of such soft skills and has to have the technology to support those (Rugarcia *et al*., 2000). This was another major problem with students' internships, and employment when they graduate. The Dept. of CSE allows its students to undergo internship program in their third year of study, i.e. just after the semester which they took the CS320 course. Different industrial organizations practice various software development process models but majority requires large amount of soft skills, team working nature and agile practices. Students who have not experienced these practices are getting hard time on the initial stages of their internships. At the same time industry prefers if the students who come as interns, to have basic understanding of the Agile practices and pair programming, in some cases.

At present the CS320 Programming Project follows milestone based project management which is more or less like to a sequential project flow activity. The major disadvantage with this approach is students blindly follow the deadline based activities without planning for their latter stage software development. They initially commit more on specification and requirement gathering activities but at the end they could not easily produce a good quality complete product. On the other hand with the present approaches, facilitation for group work or knowledge sharing on programming skills is very minimal. The environment is also vulnerable to most of the above mentioned issues since lack of flexible process model to practice. Therefore, the Agile process was selected to use as an alternative approach to overcome these problems. All these above yield the significant question on evaluating the application of Agile practice in computer programming courses. With the Agile practices' reputation on industrial projects, it ensures solutions for above mentioned problems if its impact on learning process is constructive. Therefore this study focused on that as its main research problem.

EXPERIMENT METHODOLOGY

Experiment Background

As explained briefly above, the experiment was conducted with the undergraduate students at the Department of Computer Science and Engineering, University of Moratuwa. According to the

Bachelors of Science of Engineering (B.Sc. Eng) honours curriculum for Computer Science and Engineering specialization, students have to follow two compulsory projects courses with intensified software programming learning. CS320 Programming Project is the third year software programming project which is mainly individual but allowed to form two member groups if there is a reasonable workload and scope of the selected project. The other course module is CS420 Project, which is a two semester and one term long final year group project (4 members). For this research, CS 320 Programming Project was considered. Most of the CSE learning activities are now based on University of Moratuwa e-Learning Platform, LearnOrg-MOODLE (Perera, 2009).

As Earle stressed, underpinning any approach to the design and delivery of learning resources should be a sound and clear pedagogical rationale (Earle, 2002), the experiment methodology was designed with careful attention to avoid any unnecessary drawbacks getting into the learning process. On the other hand the research was conducted during a running course module where the students learning outcomes and results could be easily affected if not conducted properly.

CS320 Programming Project is a GPA (Grade Point Average) 2.0 credit worth of course module which has 6 hours of lab works / equivalent lecturing or combination of both to form the credit value. Since this is a compulsory course module every student has to follow the course and earn the credits. Therefore 100 students were enrolled to the course module. Out of those 100 students there were 22 project proposals requesting to do a group project. However, 5 project proposals were rejected by the proposal selection committee which due to one or more of the following reasons;

- Lack of project scope for a worth of GPA 2.0 credit work
- Lack of project scope for a group credit requirement (i.e. credit 2.0 x 2 for two members)
- Infeasible project scope to complete within the time and effort constraints with the course requirements
- Insignificant component of programming compared to the project scope.

Therefore, 17 group projects were approved and other students involved in individual projects. However, only 11 groups were allowed and agreed to take part of this study due following reasons.

- The project domain constrains do not supportive enough to practice pair programming
- Low motivation to follow the new practice with backed fear of getting low marks due to new practices
- Too weak – weak or too expert – expert scenarios which can produce output either overriding the actual impact of the agile practice or impact to their final output

The Students' cumulative GPA was considered as a measure for their skill level for computing. The considered student sample had the highest cumulative GPA value of 4.06 and lowest 2.64 (out of maximum 4.2). As mentioned in the research problem, since the objective of assessing agile practice as a weak student competency improving approach the selected groups were having a reasonable skill difference between the two members. However, having more than 0.75, variance was not considered to avoid situation expert – weak scenario; none of the groups were in that kind, though. To ensure these conditions preserve, two norms were introduced for selecting groups for this study. One student should have the cumulative GPA in the range of 3.40 – 3.80 and the other should have his GPA in the range of 2.90 – 3.40. The groups' GPA distribution, Mean GPA and Skill Variances are shown in the table 1.

***Table 1**: Student GPA distribution of selected groups for the study*

Group	1	2	3	4	5	6	7	8	9	10	11
Student 1 GPA	3.52	3.67	3.54	3.55	3.71	3.69	3.7	3.75	3.78	3.61	3.58
Student 2 GPA	2.91	3.4	3.32	3.01	3.08	3.13	2.97	3.34	3.25	2.9	3.2
Group Mean GPA	**3.22**	**3.54**	**3.43**	**3.28**	**3.4**	**3.41**	**3.34**	**3.55**	**3.52**	**3.26**	**3.39**
Skill Variance	**0.61**	**0.27**	**0.22**	**0.54**	**0.63**	**0.56**	**0.73**	**0.41**	**0.53**	**0.71**	**0.38**

The entire students' cumulative GPA distribution is shown in the figure 1. The class cumulative GPA mean value μ_1 was 3.475

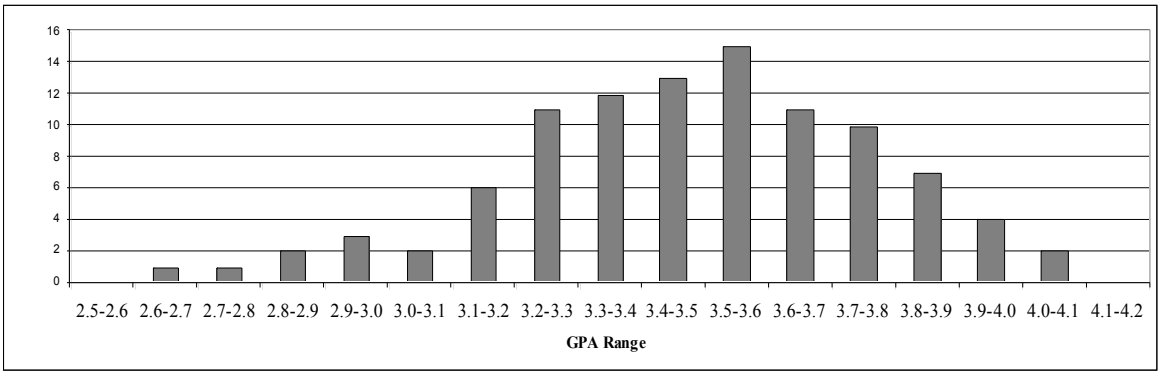

***Figure 1**: Cumulative GPA distribution of the entire student sample*

Experiment Setup

CS 320 course duration is 15 week semester. However, since the students are instructed to finalize their project ideas with approval before the semester starts, they had almost 15 weeks for their project works. Out of that 3 weeks are used to mid semester, end of semester evaluations and project report preparations. Other 12 weeks are used for requirement elicitation, high level system architectural design, low level detailed design, system development, system testing and finally working system implementation. One advantage with the experiment setup is that the students were already following the course CS 302 Software Engineering which already has covered a substantial amount of knowledge on Agile Software Practice. Therefore, the 11 groups could easily follow the practice guidelines without any difficulty; a group of expert lecturers were readily available for support any difficulty on agile practice, though.

During the 12 weeks of project work, students have to spent some time on producing project related documents, such as Fortnight Report (a report on the project progress in every two weeks), finalized Requirement specification, Design Specification with Test Plan, and finally the Project Report (a professional document following the industrial standard format on the entire project work they carried out). As per the course module evaluation requirement, except for Final Project Report, all other documents need to be submitted individually even for the group projects.

However, the group project students were allowed to share their content on the documents with their colleague group member to avoid unnecessary content replication and to preserve the consistency on what they claim as their work.

The concept of **Pair Programming** was the major change introduced to the experiment groups. Other than that the basic agile principles were also followed by the groups as the norms of the practice.

Following Pair Programming – Students were given laboratory facilities (working place for two members together) to practice the pair programming. It was a controversial change to the fixed laboratory layout which encourages individual students to use the computers alone. In this study this was the major difference experiment students experienced when compared with other students' works.

Individuals and Interactions over Processes and Tools – This was easily practiced as the students were not allowed to use any excessive tools for their development other than the IDE (Integrated Development Environments) and third party libraries with valid justification. Since they are practicing the Pair Programming they had a strong trust on other partner's commitment to their success with continues interactions.

Working Software over Comprehensive Documentation – This was a difficult practice to achieve with the experiment environment as the course module requires certain documentation. Slightly compromised approach was followed to avoid any risks on students' final grading, making a win-win situation with supervisory advices to interchange tasks time to time (coding to documenting and vice-versa)

Customer Collaboration over Contract Negotiation – In a student learning environment there is a rare chance to have an external customer to interact. As the students were assigned to a tutor/instructor with sufficient domain and programming knowledge, those were considered as the customers respectively for the students. This consideration was not significant to other students but for the groups who practiced the agile process. The students were asked to contact the resource persons frequently and accommodate their alterations collaboratively. Contract Negotiation was not applicable.

Responding to Change over Following a Plan – This is a quite tricky principle which many practitioners believe as the agile practice does not allow following a project plan. In fact what it really means is that you should have a plan of your progress; which indeed the students are encouraged to have for their all studies, but when a change is due, responding to that should not be hindered merely because of following the stages of plan. Simply it means whenever changes to be done, those should be given more priority than the project plan. In this experiment as well with the many other student activities, it is general to see that students do their development in that nature to have error free code base for next stage developments.

Following these principles and full time pair programming, students of the selected 11 groups completed their projects with meeting all the deadlines of the course. The next section describes the results and their analysis.

RESULTS AND ANALYSIS

To maintain the project evaluation fairness, the author did not involve in any kind of evaluation of the course, no evaluators were aware of this kind of study and they were asked to evaluate a uniform sample of projects. At the end of the evaluations and the results of the students been

finalized, the student grade distribution was as follows in the figure 2. Excellent project were with A+ and A grades and good projects were obtained A- and B+ grades. The grades B and B- were considered as satisfactory level and C+, C, and C- grades were on marginal pass range.

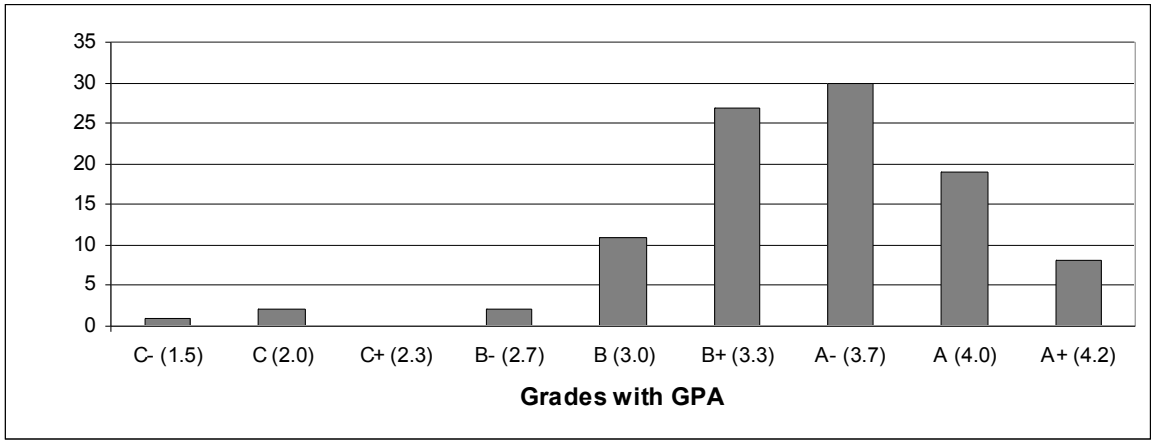

***Figure 2**: CS320 Grade distribution of the entire course (100 students)*

The above mentioned grades were given on the basis of Continuous Assessment. Therefore, the students have to obtain marks from various components. Also these components were weighted as their relative importance to the objectives of the course module. Major areas of marking and their average student marks for the students who practiced Agile principles and others are shown in the following figure 3.

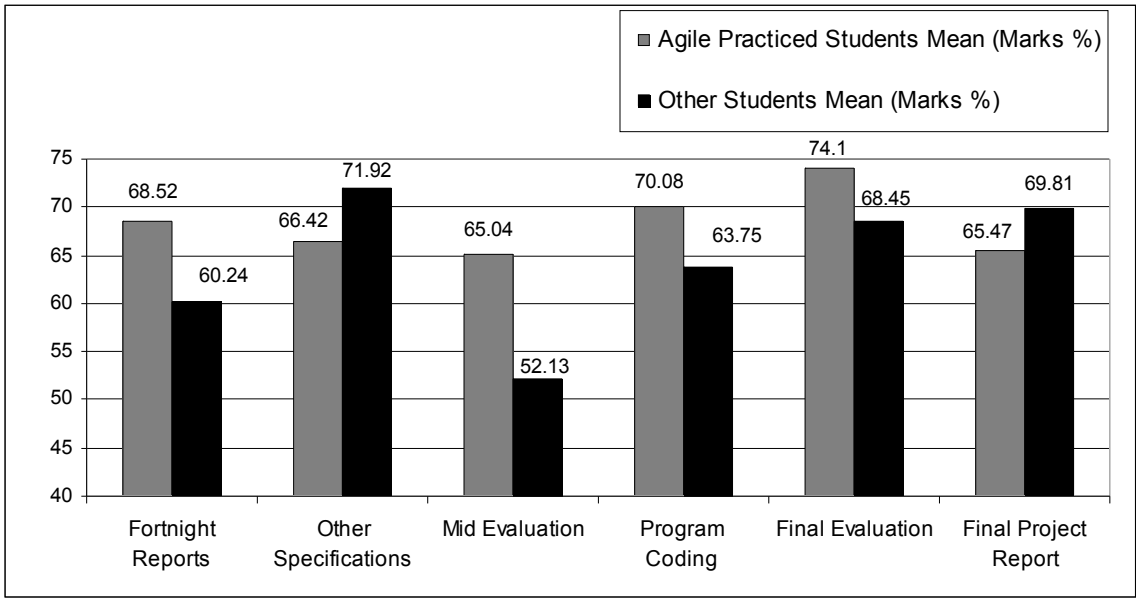

***Figure 3**: Mean marks for different components of the course between the Agile practiced students and others*

If each of the marking components analyzed thoroughly, following explanations could be derived. The Fortnight reports are as explained early, the most frequent deliverable that the students had to produce. According to the genuine Agile practices, they encourage frequent deliverable; not documents but working code, though. As in this research, to avoid any student risks of getting low marks, the slightly altered approach of producing frequent deliverables, both working code and relevant documentation in simple nature was helpful to Agile groups to score higher average mark over the others. Also the examiners were much keen on to see good progress with project through the Fortnight reports than other contents.

When Other Specifications are considered, it includes Requirement Specification, Design Specification, Testing and Implementation Plan, and usability related documents such as User Manual, Read Me texts, or Installation Guides. Agile practiced students average mark is 5.5% less than that of the other students. Similar observation can be seen when consider the Final Project Report marks too. This shows a significant issue with proper documentation by Agile projects. It was anticipated at the beginning of the projects since the Agile principles do not encourage any sophisticated documentations. Though the practiced approach was a combination of documentation with coding and Agile students could score well in Fortnight Reports, when it comes to the one time one type of lengthy documents they failed to keep their lead with others. When the course module was completed, there was a discussion session with the Agile groups to identify their concerns and feedback. Students describe their difficulty with producing detailed and sophisticated specifications about their project. Their main concern was that they are highly competent to explain their program code and its behaviour but not much the project parameters like schedules, specifications and planning activities. This is a major finding with the Agile practice to be used as a tool for teaching computer science.

On the other hand, for Program coding and Final Evaluations, Agile students scored average marks of 6.33% and 5.65% respectively over than other students. For these components, quality of the code, usability of the product, and achieving their project goals were mainly considered. Agile practise shows a significant help to students to get higher marks. The most significant observation was with the Mid Evaluation which was conducted 7 weeks after the commencement of the project. At that time almost all the Agile projects were way ahead with program coding compared to other students. The others were mainly focused too much on preparing specifications and planning their future project activities, which is not a good sign for an industrial software project that follows Rapid Development (RD). The mean marks deference is 12.91% which is very significant difference when compared with other marks. However, at the Final Evaluation this difference was reduced 5.65% with extra efforts by other students at the latter weeks merely to complete their projects before the deadline. Because of that many students who did not practice Agile methods have not got a sufficient time to consider usability aspects and fine tuning of their products.

Marking components such as Presentations, Answering to questions, and Demonstrations were not considered for this analysis as they do not shows stronger correlation with the experiment conditions than a mere personal competence on soft skills.

The table 2 shows the final grades of the 22 students for the course module while indicating the group skill variance before and after the course module. It is worth to mention that all the 11 projects were able to get either Excellent or Good grades at the end. The reason to use the skill variance of the students between their partners is important to explain an outstanding observation of from this study. As explained in the experiment information, only groups who had skill variance between 0 to 0.75 GPA were used for the study, where the minimum skill variance in the sample was 0.22 (Group 3) and the maximum skill variance was 0.73 (Group 7). When the projects were

completed, by considering their individual grades for the course module, new skill variance was derived. Interestingly, four groups out of the 11, showed 0 skill variance. The maximum skill variance reported after the study was 0.5. The average skill variance before the study μ_{SVb} = 0.508 and the average skill variance after the study μ_{SVa} = 0.209, which is a clear indication of the reduction of the skill gap of students. This effect is a promising feature of using Agile practices in teaching computer programming which is perceived as a difficult learning area by students and academia.

Group	1	2	3	4	5	6	7	8	9	10	11
Student 1 Module Results Grade & GPA	A- (3.7)	A (4)	A- (3.7)	A (4)	A+ (4.2)	A (4)	A (4)	A+ (4)	A+ (4.2)	A (4)	A (4)
Student 2 Module Results Grade & GPA	B+ (3.3)	A (4)	A- (3.7)	A- (3.7)	A- (3.7)	A (4)	A- (3.7)	A (4)	A+ (4.2)	A- (3.7)	A- (3.7)
Skill Variance Before	0.61	0.27	0.22	0.54	0.63	0.56	0.73	0.41	0.53	0.71	0.38
Skill Variance After	0.4	0	0	0.3	0.5	0	0.3	0.2	0	0.3	0.3

***Table 2**: Agile Groups Students' Individual Results for the Course with their Skill Variances before & after*

Furthermore, these students' results were analyzed to examine their respective results growth which is shown in the figure 4. The relative growth/decline was considered using the following formula.

{(CS320 Course GPA – Cumulative GPA) / Cumulative GPA}*100%

All the obtained values are positive and hence shows growth for every student subjected to the study. In this study the more competent students were named as student 1 in the groups and the relatively weaker one as the student 2. According to the analysis student 2 in each group shows a significant relative growth of their GPA for the course module against their group mates. The growth range varies from 11.45% to 29.23% for the weaker students. On the other hand competent students also reported reasonable growth of their GPA in the course module ranging from 4.52% to 13.21%. This shows that the weaker students are the most benefited people from the Agile practice. Of course it has a rational of sharing the knowledge which eventually reducing the knowledge gap between the students as observed in the above. However, when the weaker students to be equal or reasonably close with their stronger group mate's skills, they have to develop more skill level than that of the stronger student's development. One could argue that practicing Agile process makes the stronger students to hinder their skill development due to helping their partners' coding problems and becoming a burden to them. But that is not the truth as most of the stronger students have reached to their maximum possible marks/grades they could obtain from the course, i.e. A+ or A grades. And they came to those grades with having reasonably near cumulative GPA values to those grades throughout their studies. Because of that their relative growth percentage shows smaller values. However, its not possible to neglect the fact that the weaker students' programming skills could be significantly increased by allowing them to practice pair programming, which is a really difficult task to achieve using the traditional programming teaching methods, otherwise. Furthermore, Liu and Chan (2006) observed similar outcomes where they found novices' gain of knowledge is higher compared to experts' gain. Furthermore, it proves the accuracy of this research and the validity of the results, too.

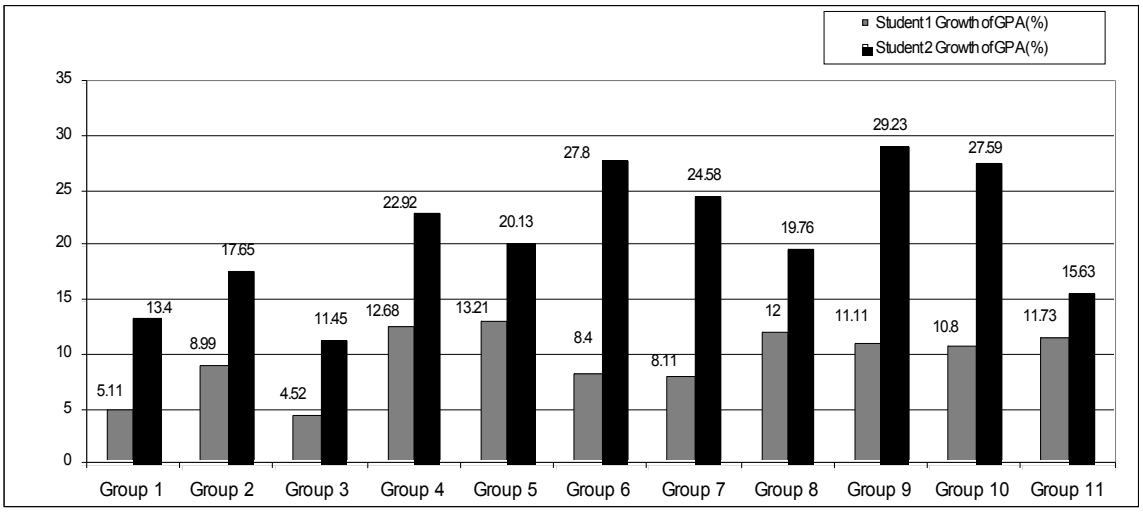

Figure 4: *Agile practiced students' course GPA growth respect to their cumulative GPA*

A simple survey was done with the 22 students who practiced the Agile practice for their project. The same survey was done before the study and after the study. In the survey, six questions were asked with one of Yes, No, and Don't Know options to be selected as the answer. The responses are shown in the table 3 below.

Questionnaire Statements	Before the Study			After the Study		
	Yes	No	Don't Know	Yes	No	Don't Know
Agile Practice is Good for Projects	5	3	14	20	2	0
It Helps to Share Knowledge	11	5	6	22	0	0
I Can be a part of industrial Agile Project Team	1	18	3	17	2	3
Recommends Agile Practice for Others	2	6	14	21	0	1
It Helps to Improve My Skills	2	5	15	22	0	0
It Helps to Raise My GPA	0	4	18	22	0	0

Table 3*: Survey results about the student perception on using Agile Process for their studies.*

The results show significant improvement of the students' perception on the Agile practice once they actually experienced it. In fact they are confident enough to accept their progress of results and skill improvement, which invariably increased their confidence on the practice. Importantly their confidence on being a member of an industrial Agile project team has increased largely. That will allow them to work well during their internship term in the industry.

Experimental Limitations and their impact

The experiment has the following limitations.

- **The user skill variance** - This is a common issue with human skill based experiments. However, as described above, various methods were used to avoid any extreme cases which can impact to the experiment significantly

- **The project domain variance** – This is yet another unavoidable concern with student programming projects. Nevertheless, there was an expert steering committee to approve the project proposals making those confined to the expected scope and work norms of the course. Therefore it is justifiable to say the project domain variance did not impact significantly, though the project differ each other.

- **The project evaluator marking variance** – Different evaluators and panels have marked the students' activities with slight differences of their marking. As the departmental practice there is a normalization process to make all the marks into a uniform platform. In addition to that the evaluation process ensured that any project should not be evaluated by the same panel more than once. Therefore the impact was minimized.

- **Academic constraints** – Since the experiment environment was a live course module there were many constraints with conflict of other courses that student took and other academic constraints which reduce the opportunity of having an industry like coding environment. However this impact was equally distributed among all the students.

When consider the entire analysis it can be summarized that the research results show important outcomes, despite these trivial yet unavoidable limitations.

CONCLUSION

The research outcomes shows significant positive impact on the student learning process by applying Agile practice for their programming works. This study took a different approach from the rest of the Agile process based studies on student learning, by focusing more on the individual student's knowledge improvement through the practice. As shown above, not only the Agile process helps to increase the students competencies on a relative difficult study area like programming, but also it does help comparatively weaker students to have a better hope on their learning without being burden to their groups. On the other hand it allows students to reach tiny issues with programming as soon as they encounter them in their work, giving a more long lasting learning experience.

There are some policy implications could be derived based on the research analyses and findings. Importantly, the results stress the point that there is immense need of improvement of teaching methods in computer science education, at least with the programming. Also, this research shows a promising future for finding more attractive and user friendly approaches for teaching programming, eliminating its technical difficulties many students experience. The author expects and encourages other researchers and scholars to extend this research with possible future studies. Also, it is a responsibility of all who involved in the education sector at policy making level to identify and implement this kind of new approaches with rational research evidences to improve the student learning experience. When that happens, all stakeholders in the learning process will be undoubtedly benefited, while making the students to experience fruitful education and a promising future.

ACKNOWLEDGEMENT

The author is thankful for the students who participated in this study with their genuine commitment to make this a success. Also the Department of Computer Science and Engineering academic staff for participating in the project evaluation process and the non-academic staff who helped on various laboratory and system issues that student came across while they were doing their projects.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., 2002, "Agile software development methods – Review and Analysis", *VTT Publications 478*, p. 17.

Agile Manifesto, 2001, "Manifesto for agile software development", [available at] http://agilemanifesto.org/ , [accessed on 07th December 2008].

Alavi, M. 1994, "Computer-mediated collaborative learning: An empirical evaluation", *MIS Quarterly* 18, pp. 159–174.

Almstrum, V. L., Hazzon, O., Guzdzial, M., and Petre, M. 2005, "Challenges to computer science education research", *In Proc. of the 36th SIGCSE Technical Symposium on Computer Science Education SIGCSE '05, ACM Press*, pp.191-192.

Arntzen, J., Krug, D., & Wen, Z. (2008, December 30). "ICT literacies and the curricular conundrum of calling all complex digital technologies Tools". *International Journal of Education and Development using ICT* [Online], 4(4). Available: http://ijedict.dec.uwi.edu/viewarticle.php?id=571 [accessed on 11th February 2009].

Arbelaiz, A. M. and Gorospe, J. M. C. 2009, "Can the grammar of schooling be changed?", *Computers & Education*, vol. 53, pp. 51–56

Bipp, T., Lepper, A., and Schmedding, D. 2008, "Pair programming in software development teams – An empirical study of its benefits, *Information and Software Technology*, 50, pp. 231–240.

Birch, D., & Sankey, M. (2008, January 30). "Drivers For and Obstacles To the Development of Interactive Multimodal Technology-Mediated Distance Higher Education Courses". *International Journal of Education and Development using ICT* [Online], 4(1). Available: http://ijedict.dec.uwi.edu/viewarticle.php?id=375 . [accessed on 26th January 2009].

Cohen, D., Lindvall, M., Costa, P. 2003, "A State of the Art Report: Agile Software Development", *Data and Analysis Center for Software* 775 Daedalian Dr. Rome, New York 13441- 4909, p. 01.

Covington, D., Petherbridge, D., & Egan Warren, S. 2005. "Best practices: A triangulated support approach in transitioning academic to online teaching", *Online Journal of Distance Learning Administration*, vol. 8, no. 1. Available online at: http://www.westga.edu/%7Edistance/ojdla/spring81/covington81.htm , [accessed 23 December 2008].

Culp, K.M., Honey, M. and Mandinach, E. 2005, “A retrospective on twenty years of education technology policy”, *Journal of Educational Computing Research* vol. 32 (3), pp. 279–307.

Dagnino, A., 2002, “An evolutionary life-cycle model with agile practices for software development at ABB”. *In Proc. on 8th IEEE Conference on Engineering of Complex Computer Systems (ICECCS’02)*, pp. 215–223

Davis, E. A., Linn, M. C., and Clancy, M., 1995, “Learning to use parentheses and quotes in Lisp”, *Computer Science Education*, vol. 6, pp. 15-31.

Dawson, R.J., Newsham, R.W. and Kerridge, R.S., 1992, “Introducing New Software Engineering Graduates to the ‘Real World’ at the GPT Company”, *Software Engineering Journal*, vol.7 (3), pp. 171-176.

Domingue, J. and Mulholland, P., 1997, “Teaching Programming at a Distance: The Internet Software Visualization Laboratory”, *Journal of Interactive Media in Education*, 97 (1), pp. 1-31.

Earle, R.S., 2002, “The integration of instructional technology into public education: promises and challenges”, *Educational Technology Magazine*, vol. 42, no. 1, pp. 5-13.

Gatfield, T. 1999, "Examining Student Satisfaction with Group Projects and Peer Assessment". *Assessment and Evaluation in Higher Education*, Vol. 24(4), pp. 365-377.

Gunter, G. A. 2007, “The Effects of the Impact of Instructional Immediacy on Cognition and Learning in Online Classes”, *International Journal of Social Science*, vol. 2 no. 3, pp. 196-202.

Harden, R.M. and Crosby, J., 2000, “AMEE Guide No 20: The good teacher is more than a lecturer-the twelve roles of the teacher”. *Medical Teacher*, vol. 22(4), pp. 334–347.

Highsmith, J., Cockburn, A., 2001, “Agile Software Development: The Business of Innovation”, *IEEE computer*, vol. 34, pp.120-127.

Johnston, L, and Miles, L, 2004, “Assessing contributions to group assignments”, *Assessment and Evaluation in Higher Education*, vol. 29 (6), pp. 751–768.

Kember, D., 1997, “A reconceptualisation of the research into university academics conceptions of teaching”, *Learning and Instruction*, vol. 7(3), pp. 255–275.

Knight, C., Knight, B., & Teghe, D. (2006, May 31). Releasing the pedagogical power of information and communication technology for learners: A case study. *International Journal of Education and Development using ICT* [Online], 2(2). Available: http://ijedict.dec.uwi.edu/viewarticle.php?id=167 [accessed on 03rd April 2009].

Lui, K. M. and Chan, K. C. C. 2006, “Pair Programming Productivity: Novice-novice vs. expert-expert”, *International Journal of Human Computer Studies*, vol. 64, pp. 915–925.

Mason, R., and Weller, M. 2000, “Factors affecting students’ satisfaction on a web course”, *Australian Journal of Education Technology*, vol. 16(2), pp. 173-200.

McCarthy, B. and Hawking, P., 2002, "Teaching SAP's ABAP Programming Language to IS Students: Adopting and Adapting Web-based Technologies", *In Proc. of IS2002*, pp. 995-1000.

Perera G.I.U.S. 2009, "Key Success Factors for e-Learning Acceptability: A Case Based Analysis on Blended Learning End-User Experience", *In Proc. of IEEE International Advance Computing Conference, IACC'09*, pp. 2379-2384

Perera, G.I.U.S. and Fernando, M.S.D. 2007, "Bridging the gap – Business and information systems: A roadmap", *In Proc. of 4th ICBM conference*, pp. 334-343.

Perera, G.I.U.S. and Fernando, M.S.D. 2007, "Enhanced Agile Software Development — Hybrid Paradigm with LEAN Practice, *In Proc. of 2nd International Conference on Industrial and Information Systems, ICIIS 2007, IEEE*, pp. 239 – 244.

Perera, G.I.U.S. and Fernando, M.S.D. 2009, (*In Press*) "Rapid Decision Making for Post Architectural Changes in Agile Development – A Guide to Reduce Uncertainty", *International Journal of Information Technology and Knowledge Management, Serial Publishers.*

Rugarcia, A., Felder, R.M., Woods, D.R., and Stice, J.E. 2000, "The future of engineering education. I. A vision for a new century", *Chemical Engineering Education*, vol. 34 (1), pp. 16–25.

Vik, G.N. 2001, "Doing more to teach teamwork than telling students to sink or swim", *Business Communication Quarterly*, vol. 64 (4), pp. 112–119.

Williams, L., McDowell, C., Nagappan, N., Fernald, J., and Werner, L. 2003, "Building pair programming knowledge through a family of experiments", *In Proc. of International Symposium on Empirical Software Engineering, ISESE 2003,* pp. 143-152

Williams, L., Yang, K., Wiebe, E., Ferzli, M., and Miller, C. 2002, "In Support of Pair Programming in Introductory Computer Science Course", *Computer Science Education*, vol. 12(3), pp. 197–202.

Original article at: http://ijedict.dec.uwi.edu//viewarticle.php?id=755