Using a Team-Based Approach in an IS Course: An Empirical Study

Glen Van Der Vyver and Michael Lane
University of Southern Queensland, Toowoomba, Australia

vandervy@usq.edu.au lanem@usq.edu.au

Executive Summary

Adaptive and fluid applications development methodologies such as Prototyping, RAD, FAD and Extreme Programming have emerged in recent years in response to organisational realities that include rapid change, uncertainty and ambiguity. These methodologies are well suited to the team-based approach that has become so important in the modern organisation. Yet, many educational programmes in the West still focus on individual learning and assessment. This paper reports on a pilot study where team-based methods are incorporated into a demanding undergraduate IS (Information Systems) course. We review some of the new approaches to applications development, in particular team-based approaches and examine how they are highly relevant to the way business is conducted today.

The study involves students who were enrolled in an undergraduate course in database development. A learning environment that incorporates elements of the "real world" and a strongly group-focused approach was designed. Although the course is relatively 'technical', it is a core course for many programmes and therefore attracts enrollments from students with varying levels of prior technical knowledge and ability. Based upon the results of diagnostic tests, GPA and a survey, students were allocated to groups that were specifically designed to foster group learning. Each group had a balance of skills and included at least one person identified as a lead programmer, who had scored 85% or above in the test, had a high GPA's and exhibited confidence in overall and course-specific technical abilities. Groups were required to complete a variety of tasks relating to the creation and maintenance of databases and database programming.

The group-based had a most promising influence on performance, particularly for those students who came from less technical background and / or had struggled in programming courses previously. People with limited technical knowledge and/or an average amount of technical aptitude do benefit from working in small teams with people who are technically strong. Furthermore, these benefits seem to extend to performance indicators that are related to the group task.

A number of problems emerged, however:

Students expressed lower levels of overall satisfaction and made a number of negative comments about the new innovations. We are also not sure that lead programmers gained as much as the others from the experience.

The course was unexpectedly demanding for the lecturer. These demands included additional preparation and consultation time, dispute resolution, mentoring and sorting out problems related to group dynamics. A course of this nature demands a significant project and there were complaints about workload. In some groups, not everybody made a satisfactory contribution.

The original version of this paper was published as one of the 24 "best" papers in the proceedings of the 2003 Informing Science and IT Education Conference in Pori, Finland http://2003.insite.nu

Keywords: Organisational Learning; Team Learning; Prototyping; RAD; FAD; Extreme Programming; Teaching IS; IS Development

Introduction

Individualism dominates western cultures and thinking. It therefore also dominates the teaching approach applied in most Western universities, where individual and critical thinking is encouraged and individual assessment is the norm (Triandis, 1995). We argue in this paper that collective thinking within a team based approach to teaching IS development can be an effective tool for student learning. Our central hypothesis is that a team based learning approach to IS development that encourages collective thinking and learning is, depending on the circumstances, more effective than an approach that focuses on and encourages individualism. A team-based approach encourages the development of skills necessary to succeed in a commercial world where the team is often more important than any individual (Argyris, 1993; Peters, 1993). In using such approach we adapted some of the central tenets of methodologies that promote team based collective thinking in IS development. Our approach attempts, as far as is practical, to integrate group techniques, such as "extreme programming", into the teaching and learning situation. The findings generally validate such an approach but also reveal a number of potential problems. We argue that IS educators should be exploring more innovative ways in which to integrate team-based learning activities into IS teaching while at the same time ensuring that high quality standards of assessment are maintained.

This paper reports on a pilot study we conducted to integrate team oriented approaches into an undergraduate course in database development. We also attempted to replicate some elements of the modern work environment into the course, for example short and tight deadlines and shifting requirements.

The structure of this paper is as follows: First, a general approach to teaching IS development as a discipline is discussed. Then, we look at current IS development approaches, such as prototyping, rapid applications development (RAD), frantic applications development (FAD) and extreme programming (XP). In particular we are interested in the concept of paired programming, which encourages a team-based collective thinking approach to system development. Then, we introduce our central hypothesis that a team-based collective approach to teaching IS development can have advantages over situations where the individual works in isolation. We outline the method we employed to test our approach using undergraduate students in a database development course. The findings of our study are presented and discussed. Finally, some preliminary conclusions are made about the effectiveness of a team-based approach to teaching IS development.

General Approach to Teaching IS Development

Western thinking is primarily centered on individualism as opposed to the collectivist thinking of many Eastern cultures. Triandis (1995) proposed that collectivism and individualism be conceptualized as polythetic constructs. An individualistic culture is one in which the individual is the focus of activity and meaning in contrast to collectivistic cultures, which are characterised by the primacy of the group (Robbins, 1997). The individual's self identity is largely dependent on the sense of belonging to groups, which can include the extended family**, teams**, neighbourhoods or tribes. Success in individualist cultures is perceived to be dependent upon hard work and the commitment of the individual to achieve. By contrast, in a collectivist culture success is more often dependent upon inter-group and interpersonal connections. Table 1 illustrates the differences between individualism and collectivism.

Individualism (representative of prevailing culture in Western Universities)	Collectivism (Representative of many Eastern/Asian cultures)
Fostering interdependence and individual achievement	Fostering interdependence and group success
Promoting self expression, individual thinking, personal choice	Promoting adherence to norms, respect for authority/elders, group consensus
Associated with egalitarian relationships and flexibility in roles (e.g. upward mobility)	Associated with stable hierarchical roles (dependent on gender, family background age)?
Understanding the physical world as knowable apart from its meaning for life	Understanding the physical world in the context of its meaning for human life
Associated with private property, individual ownership	Associated with shared property, group ownership

Table 1 Differences and salient features of individualism and collectivism (adapted from Trumbull, Rothstein & Greenfield 2000, p. 1)

Most universities in the West teach the Information Systems discipline from an Information Systems Development perspective, with an emphasis on software development (programming skills), network management, project management and the systems development life cycle. While there is often some team based learning in one or more capstone courses, there is a predominance of emphasis on individual learning. We do not argue against individual thinking as it is important and valuable in its own right. Individualism in the Western World has positioned the West well in the emerging knowledge based, networked global economy where creativity, innovation and entrepreneurship have been of crucial importance.

Collective thinking, however, which emphasizes the importance of team based learning, does have value in IS development. The discipline of IS has long been characterised by the rapid pace of change in ICT technologies and the challenges these present. New systems development methods have emerged to accommodate these challenges (Tudhope, Beynon-Davies & Mackay, 2000; Yourdon, 2000). While prototyping and RAD are well established newer, more "extreme" methodologies have emerged. These include FAD and Agile Methodologies (Fowler 2000; Yourdon, 2000). They address the new corporate imperatives in the Age of the Net: the need to develop information systems in "Internet Time" to take advantage of new and emerging markets, short windows of opportunity, tight profit margins and competitors who are anywhere and everywhere (Gordon & Bieman, 1996; Highsmith & Cockburn, 2001; Martin, 1991).

It is also perhaps paradoxical that our students work on many of their courses in isolation and then join corporations where teamwork is highly prized. The vision of the team-based corporation is now a reality (Argyris, 1993; Peters, 1987, 1993). It is now the norm for knowledge workers to move from team to team (and role to role), always adapting, always learning. In situation such as these, team learning plays a crucial role. Many of today's problems are so complex, their information content so overwhelming, that a team approach is essential.

Methodologies for the Internet Age

Two emergent methodologies, FAD (Frantic Application Development) and Agile, seem to be best suited to the extraordinary demands of the Internet age. Yourdon (2000) argues that RAD (Rapid Application Development) cut development times from years to months but this is not enough for the Internet

Age. In the chaos that is development for the Internet, development time is cut from months to weeks to days. Yourdon concedes that FAD is practitioner driven. Agile methodologies have a more solid theoretical basis at this juncture. Extreme programming (XP) is the most popular agile methodology (Fowler 2000). In this approach, there is a strong emphasis on the team and collaborative development. While there is knowledge transfer between the customer and developers very much in line with the RAD approach, XP goes further and adopts a collaborative approach within the development team by using the "whole team" approach and "paired programming."

Agile methodologies (which include XP) and FAD, therefore, champion the cause of team-based and collaborative approaches in contrast to the more traditional IS development methodologies. At the core there are four key values (Agile Alliance Manifesto, 2002):

- Individuals and interactions are valued above processes and tools.
- Working software is valued above comprehensive documentation.
- Customer collaboration is valued above contract negotiation.
- Responding to change is valued above following a plan.

It is clear the Agile Methodologies have much in common with the RAD approach. These methodologies are best employed when rapid adaptation to emerging situations is required. They are not ideal for the development of major, mission-critical systems. The question arises, "Are these methodologies really any different to the RAD approach, with its focus on incremental prototyping, JAD sessions and teamwork?" We would argue that the approach is different because it approaches the core act of systems development, the creation of code, in a radical new way. In a RAD team, much of the work is done in the team context but, when all the talking and live development of prototypes is over, individual programmers often return to their individual workstations and work alone on specific components of the project. The XP approach also has a much stronger focus on team learning.

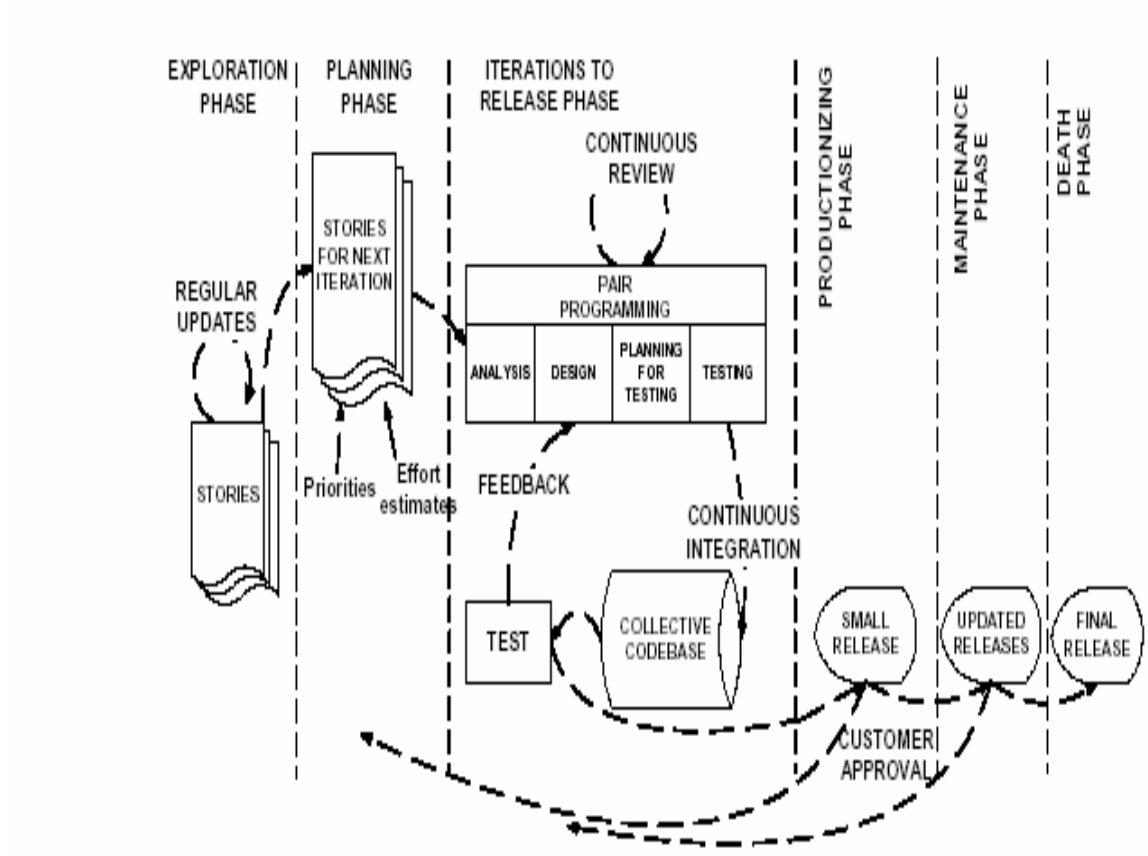
The XP methodology has 12 core practices (Beck 2000):

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Changing requirements are welcomed, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Software is delivered quickly and frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Projects are built around motivated individuals. Give them the environment and the support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 1 illustrates how these twelve core practices of XP fit into the exploration, planning, iterations to release, product ionizing, implementation and death phases of the system development life cycle (Abrahamsson, Salo, Ronkainen, & Wasta, 2002). The terms used for the various phases of system life cycle in the XP approach, although somewhat different, still emphasize the standard analysis, design, build, implementation, maintenance phases in an ongoing cycle until the ultimate death of a system. However, in the XP approach there is an emphasis on priority of customer/client stories that represent the business requirements for each system release. The involvement of customer from the inception of a project through to the Customer/client acceptance testing before production release of code ensures strong buy in by the Customer/client.

Figure 1 Different activities of XP associated with the phases of the system development life cycle (source: adopted from Abrahamsson, Salo, Ronkainen & Warsta 2002, p. 19)



Team Based Approaches to IS Development

Whole Team

All contributors in an XP project sit together as members of one team (Abrahamsson et al., 2002). The team must include a business representative client/customer who provides the requirements and sets the priorities and direction of the project. It is important that the customer involved in a XP project is a real end user who knows the system domain and what is needed. The team will consist of programmers and system testers who will help the customer define the acceptance tests. Business analysts will be involved helping the customer to define the requirements. There is commonly a coach who helps keep the team on track and facilitates the process. There may be a manager, whose role is providing resources, handling external communication and coordinating activities. One or more of these roles may be the responsibility of one individual. Everyone on an XP team contributes in any way they can. The best teams have no specialists - only general contributors with special skills.

Pair Programming

All production software in XP is built by two programmers, sitting side by side, at the same machine (Abrahamsson et al., 2002). This practice ensures that all production code is reviewed by at least one other programmer, and results in better design, better testing, and better code. It may seem inefficient to have two programmers doing "one programmer's job", but the reverse is true. Research into pair programming has shown that pairing programming while producing less code per programmer, more importantly, produces better quality code with fewer bugs (Williams, 2000). It is a somewhat controversial practice that has met some resistance from programmers that are used to working independently. It is important to pair the right blend of personalities. It does take practice and time to do well and produce results. Paired programming, in addition to providing better code and testing, also serves to communicate knowledge throughout the team. As pairs switch, everyone gets the benefit of the specialised knowledge that exists within a team. Programmers learn from other programmers in the team, individuals' skills improve and they become more valuable to the team. The following comment by Nosek illustrates the potential positive outcomes of team based paired programming approach of XP: "To the surprise of managers and participants, all of the teams outperformed the individual programmers, enjoyed the problem solving process more, and had greater confidence in their solutions" (Nosek, 1998, p. 109).

This study addresses a particular aspect of XP and FAD, and that is the extent to which technical knowledge can be passed on within the team context, particularly where time and performance demands on the team are relatively challenging. We view technical knowledge as more than the ability to produce good code. Ideally, by watching a technically superior person produce good code we are also observing the solution of business problems. We learn about the business itself and about the process of solving problems.

Research Questions

This study was designed to test empirically the proposition:

A team based collective thinking approach to learning IS development in an undergraduate course is more effective than an approach based on individual effort.

We also examine some of the problems encountered and issues raised by a team based collective thinking approach to learning IS Development in an undergraduate course. The results of this study will be used to further enhance and develop curriculum development initiatives.

The study involves students enrolled in a course in database development. We believe that this course is highly suitable for the study because it is a core course in several programmes offered by the Depart-

ment of Information Systems. It therefore attracts students from a variety of backgrounds, specifically those specialising in a technical Computer Software Development (CSD) major and those specialising in Information Technology Management (ITM). The course combines database theory, programming and technical issues. Students enrolled in the Computer Software Development major have generally had good pass rates in the course. For students enrolled in the Management streams, this course has, historically, been a major challenge. The database course is by far the most technical course the ITM students are expected to take. Although these students have, nominally at least, all completed an introductory programming course, there is significant evidence to indicate that many find the course stressful and very difficult.

Given that databases are so crucial to modern IT, we are of the opinion that the course should be passed by all our graduates with no compromise as to quality and level of difficulty. It is also virtually impossible to split the ITM and CSD groups, for reasons of equity as much as scheduling. The course has, in the past, involved a compulsory practical test and a compulsory examination. Failure in either element entails failing the course. The IT Management group have consistently not performed as well CSD group in the course as a whole and, in particular, on the practical test.

This study attempts to find a way of transforming the weakest point of the course into its strongest point by adopting some of the team based techniques described above. By incorporating elements of whole team and pair programming into the course, we hypothesize that levels of performance and satisfaction with the course will improve across the board. The design of the course was also informed by recent research in the area of peer driven learning and team learning (Cardno, 2002; Cracolice & Deming, 2001).

We use data from the 2002 cohort, who were exposed to the new methods, and the 2000 cohort, who were not. The 2000 cohort were assessed by means of a practical test and an examination. They did no group work. The 2000 cohort comprised 18 IT Management students and 39 Computer Software Development students while the 2002 cohort comprised 60 IT Management students and 34 Computer Software Development students.

Hypotheses that will be Tested

Information Technology Management Group (ITM)

Hypothesis 1:

H_0: The level of performance on a practical test for a group of ITM students that has worked on a project using agile techniques and a group of ITM students that has not worked on a group project is the same.

H_1: The level of performance on a practical test for a group of ITM students that has worked on a group project using agile techniques is higher than the level of performance of a group of ITM students that has not worked on a group project.

Hypothesis 2:

H_0: The level of performance on the course overall for a group of ITM students that has worked on a group project using agile techniques and a group of ITM students that has not worked on a group project is the same.

H_1: The level of performance on the course overall for a group of ITM students that has worked on a group project using agile techniques is higher than the level of performance of a group of ITM students that has not worked on a group project.

Hypothesis 3:

H_0: The level of course satisfaction for a group of ITM students that has worked on a group project using agile techniques and a group of ITM students that has not worked on a group project is the same.

H_1: The level of course satisfaction for a group of ITM students that has worked on a group project using agile techniques is higher than the level of course satisfaction of a group of ITM students that has not worked on a group project.

Computer Software Development Group (CSD)

Hypothesis 4:

H_0: The level of performance on a practical test for a group of CSD students that has worked on a project using agile techniques and a group of CSD students that has not worked on a group project is the same.

H_1: The level of performance on a practical test for a group of students that has worked on a group project using agile techniques is higher than the level of performance of a group of students that has not worked on a group project.

Hypothesis 5:

H_0: The level of performance on the course overall for a group of CSD students that has worked on a group project using agile techniques and a group of CSD students that has not worked on a group project is the same.

H_1: The level of performance on the course overall for a group of CSD students that has worked on a group project using agile techniques is higher than the level of performance of a group of CSD students that has not worked on a group project.

Hypothesis 6:

H_0: The level of course satisfaction for a group of CSD students that has worked on a group project using agile techniques and a group of CSD students that has not worked on a group project is the same.

H_1: The level of course satisfaction for a group of CSD students that has worked on a group project using agile techniques is higher than the level of course satisfaction of a group of CSD students that has not worked on a group project.

Research Method

An experiment was designed to collect data to test the research hypotheses. The groups are independent and we assume a normal distribution. We acknowledge that independence cannot be assumed for statistics where two groups within a single cohort is involved and conduct no analysis within cohorts for the purposes of this study. We therefore use a t-test for the comparison between two means to test all the hypotheses (Zikmund, 2003). We concede that differences in numbers across the two cohorts could influence the results and we offset this by using a sensitive level of significance.

Participants in the study comprised two classes of students enrolled in a Database Development course. There was no random assignment. The class of 2000 was not exposed to a group project or agile methods. The class of 2002 was exposed to the new approach, as described below:

1. Students in the course attended a lecture, tutorial and intensive, 2-hour laboratory session each week. All students were allowed to choose laboratory sessions according to individual preference but there was a small amount of manipulation to ensure that the IT Management group was well

dispersed through the practical sessions. The assessment included a group project, an individual practical test and a formal examination.

2. The first four weeks of the course were devoted to an intensive introduction to SQL. Prior exposure to the language ranged from high to minimal. During this period, each student worked independently. Laboratory sessions involved developing rapid solutions to a substantial numbers of problems.
3. At the end of the four week period, all students took a diagnostic test. This test was compulsory and comprised a minor element of the overall assessment. During the test session, each student also completed a small survey related to that student's level of perceived competence and predicted performance in the course.
4. Students were allocated to groups on the basis of their test score, overall GPA and their answers to survey questions. Each group had four members.
5. For each group, we assigned a lead programmer. Lead programmers were all technically strong, scored 85% or above in the test, had high GPA's and exhibited confidence in their overall and course-specific technical abilities.
6. Each group was required to complete a group project. A bad failure in the project (40% and below) entailed automatic failure of the course.
7. The group project involved designing a small database, creating the database structures, writing procedural SQL programs, stores procedures and triggers and, finally, creating basic GUI Forms and Reports.
8. Although students were provided with an overall objective for the project, tasks were assigned on a weekly basis, tying in with the material studied that week. Many tasks involved the solution of problems and some tasks required students to revise and review work already completed.
9. During the two-hour practical session, the lead programmer would "cut code" while the others watched and offered comments. During the practical, the lecturer spent time with each group, answering queries, offering suggestions and helping groups develop problem-solving skills. The lecturer rarely, if ever, provided a direct answer to a technical problem. Rather, ways in which the group might find the answer themselves were explored. Each group met privately for a further two hours each week. As the volume of work was relatively large, the groups would usually split into two for the second session, with the student identified as the second best programmer in the team taking charge of the second group. Each group member would then complete additional tasks on their own.
10. At the end of the semester, each group demonstrated their small system and provided a report specifying the allocation of work. Marks were awarded on the basis of the project as a whole and in terms of the contribution of each individual group member. During the final week of semester, all students took a two-hour practical test in the laboratory. The test involved the construction of a small system using the techniques learned during the group sessions. Directly after the test, each student demonstrated their work to the lecturer.
11. Each completed a feedback questionnaire. One of the questions was worded as follows: In percentage terms (0-100), rate your overall level of satisfaction with the content, presentation and work structure of this course. A section was also made available for free format comments.

Findings

Our findings are summarised in Table 2.

The findings show that CSD students consistently perform better than ITM students on the practical test and in the course as a whole. When team based methods are introduced into the course, this effect is

Cohort	2002			2000		
GROUP	Practical Test	Overall Mark	Level of Satisfaction	Practical Test	Overall Mark	Level of Satisfaction
IT Management (ITM)	73.17	69.8	80.03	57.32	63.05	77.4
Software Development (CSD)	80.61	71.4	88.36	73.3	68.1	91.04

Table 2: Average scores (%) for Practical Test, Course and Level of Satisfaction

moderated but by no means extinguished. This is not surprising, as the CSD group have much more technical exposure than the ITM group. There is an interesting perceived positive correlation between mark levels and satisfaction levels.

We examine the six hypotheses that were tested:

Hypothesis 1:

Reject the null hypothesis, with t=21.59 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. The ITM students exhibited a dramatic improvement in level of performance for the practical test. The average increased by more than 13% which was beyond our expectations.

Hypothesis 2:

Reject the null hypothesis, with t=10.88 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. The ITM students exhibited a sound improvement in level of performance for the course overall. While some of this must be attributed to improved performance in the practical test, some of the variance in performance is likely to be attributable to improved marks for the project and examination.

Hypothesis 3:

Reject the null hypothesis, with t=3.811 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. Levels of satisfaction improved for ITM students but the margin of increase was significantly lower than the increase in mark levels. The t test statistic is clearly not as decisive here and it is possible that the variance is attributable to increase in marks alone rather than to the course content. We had expected levels of satisfaction to improve more significantly in response to the amount of effort that was put into making the course informative, interesting and challenging.

Hypothesis 4:

Reject the null hypothesis, with t=8.99 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. Although not as dramatic as the improvement shown by ITM students, the performance level of CSD students also made significant gains in the practical test, to the extent that we are a little concerned about over performance. The average increased by 7%, which was greater than we envisaged. It would appear that weaker CSD students also benefited from working with strong lead programmers.

Hypothesis 5:

Reject the null hypothesis, with t=5.14 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. The CSD students exhibited a small improvement in level of performance for the course overall. We are somewhat concerned that this variance could be largely attributed to improved marks for the project and are therefore reluctant to accept this outcome without further investigation.

Hypothesis 6:

Accept the null hypothesis, with t= -3.9411 in the one-tailed T distribution with 60 d.f. and level of significance = 0.005. Levels of satisfaction improved for CSD students but the margin of increase was significantly lower than the increase in mark levels. It should be said that overall levels of satisfaction for CSD students were already high, and it is possible that there was simply not much room for a dramatic increase.

Free Format Comments

Although we are not evaluating these in detail for the purposes of this study, a number of interesting comments were made that have a direct bearing on what we are attempting to measure in the study. We have completed a preliminary analysis and have found significant numbers of students raised points about the following issues:

1. Workload

A number of students expressed the opinion that the practical work component of the course was too large. Some argued for a reduction of up to half of the amount of work required. It would appear that the overall consensus was that the amount of work was significantly above average for courses at this level. Much of this additional work was associated with the group project.

2. Group problems

Some typical group dynamic problems occurred. In most cases, this related to group members who were not "pulling their weight." In one particular instance, this accusation was levelled at the lead programmer and a significant amount of effort was required to provide the group with a new leader and appropriate support. Other problems included lead programmers who were too dictatorial, insensitive or difficult to understand. Group problems did take a significant amount of lecturer time.

3. Tasks not specified "up front"

There was a surprising amount of complaint that the group tasks were not specified in totality at the beginning of the semester. Students seemed to discount the fact the course is attempting to replicate elements of the "real-world."

Conclusions

The introduction of the new structure into the practical element of the course had a most promising influence on performance, particularly for those students who had struggled the most before. It is clear that people with limited technical knowledge and/or an average amount of technical aptitude do benefit from working in small teams with people who are technically strong. Furthermore, these benefits seem to extend to performance indicators that are related to the group task.

We therefore come to the conclusion that a team based collective thinking approach to learning IS development in an undergraduate course is more effective than an approach based on individual effort. That being said, we are cautious about making broad statements. As this study is ongoing, longitudinal data should either support and confirm our central hypothesis or reject it. We were surprised that stu-

dents expressed lower levels of satisfaction and made a number of negative comments about the new innovations. We are also not sure that lead programmers gained as much, if anything, from the experience. A key problem seems to be the definition of effectiveness. Perhaps improved grades alone are not enough. This study indicates that we need to think carefully about this construct. Certainly, we need to consider that one student's effectiveness may be another student's frustration.

Clearly, based on this "laboratory study", techniques based on the theories of Agility, FAD and RAD have significant potential within the context of organisational learning and the performance levels of programming teams. While this type of study can never replicate the environment of a programming team working on a "real" project, we believe that the indicators for improvement may cross the divide from academia. A key finding relates to the acquisition of specific technical skills. Whereas, in the past, there were a number of ITM students who never really managed to master the core elements of the programming, there were few such students in 2002. The practical test is highly correlated with the group work and the very small failure rate and high average for that test indicates that many students were able to gain a sound knowledge of the technical skills within their group.

We are therefore encouraged by these results and plan to use the method again next semester. As we structure the new presentation, we will need to address a number of issues that have emerged from this study or could not be included due to the essentially "pilot" nature of the project. These considerations include:

1. Lecturer time

This course is resource intensive. Preparing materials accounts for many hours but mentoring groups can take even more time. Practical sessions are intensive and intellectually demanding. Unpredictable problems arise frequently and there are the perennial group dynamics issues to consider. Furthermore, mentoring the groups requires a high level of technical and theoretical knowledge. Running the course requires significant commitment from the teaching team but the results have been rewarding.

2. Student issues regarding workload, overall satisfaction and problem specification

A significant number of students are unhappy with the workload and this will need to be addressed from a psychological perspective. We are convinced that the course cannot be run using a trivial project. Students need to spend time working on complex, mutating problems if this course is to be effective. We will therefore need to develop methods to enthuse and motivate students.

3. Measures of effectiveness.

We need to develop a wider set of measures (see above).

4. Hidden effects

Some of the course assessments were not included in this study. Although these assessment types were controlled across the two cohorts (e.g. exam) we need to further examine their possible influence.

5. Informal groups

It is likely there were pre-existing informal groups and that others formed during the semester. It is very difficult to trace their influence.

6. Mark inflation

Some of the improvements in performance were so dramatic that we are concerned about mark and grade inflation. The question arises whether there was some factor or factors influencing the 2002 cohort

that biased the results. We are of the opinion that further study will be required to validate this study in terms of this concern.

In summary, we would argue that there is some indication that the techniques we used do have promise in formal and informal learning situation. We also feel there is the additional benefit of exposing students to team based collective thinking IS development. Such an approach prepares them for the commercial reality where IS development is very much team based and requires collective thinking as information systems are not developed in isolation. We intend to continue improving the way in which we use these methods. Perhaps the greatest challenge to success of this innovative approach to teaching IS development relates to the amount of resources required to run the course and the perceptions of students. It is perhaps one of these problems, rather than the issues of performance, that will in the longer term force us to return to a more standard course format.

References

Abrahamsson, P., Salo, O., Ronkainen, J. & Wasta, J. (2002). *Agile software development methods: Review and Analysis*. VTT Publications.

Alavi, M. (1984). An assessment of the prototyping approach to information systems development. *Communications of the ACM, 27* (6), 556-573.

Agile Alliance Manifesto (2002). Retrieved 10 December 2002 from: www.agilealliancegroup.com

Argyris, C. (1993). *On organizational learning.* Cambridge, MA: Blackwell.

Beck, K. (1999). *Extreme programming explained*. New York: Addison-Wesley.

Cardno, C. (2002). Team learning: opportunities and challenges for school leaders. *School Leadership and Management, 22* (2), 211-224.

Carey, T. & Mason, R. (1983). Information systems prototyping. Techniques, tools and methodologies. *INFOR, 21* (3), 177-191.

Cracolice, M. & Deming, J. (2001). Peer-led team learning. *Science Teacher, 68* (1), 20-25.

Fowler, M. (2000). *The new methdology*. Thoughtworks.

Gordon, V. and Bieman, J. (1996). Rapid Prototyping: Lessons Learned. *IEEE Software,* 30(1), pp. 85-95.

Highsmith, J. & Cockburn, A. (2001, September). Agile software development: The business of innovation. *Computer,* pp. 120-122.

Johnson, D. & Johnson, R. (1999). Making cooperative learning work. *Theory into Practice, 38* (2), 67-73.

Martin, J. (1991). *Rapid application development*. New York: Macmillan.

Nosek, J. T. (1998). The case for collaborative programming. *Communications of the ACM, 41* (3), 105-108.

Peters, T. (1982). *Thriving on chaos*. London: Pan.

Peters, T. (1993). *Liberation management*. London: Pan.

Robbins, S. A. (1997). Implications of distance education as an agent of sociocultural change. *Educational Multimedia and Hypermedia*, pp. 1791-1798.

Triandis, H. (1995). Individualism and Collectivism. In: *New Directions in Social Psychology*. Colorado: Westview Press.

Trumbull, E., Rothstein-Fisch, C. & Greenfield, P. M. (2000). *Bridging cultures in our schools: New approaches that work.* A WestEd Knowledge Brief.

Tudhope, D., Beynon-Davies, P. & Mackay, H. (2000). Prototyping praxis: Constructing computer systems and building belief. *Human-Computer Interaction, 15*, pp. 353-383.

Williams, L., Kessler, R. R., Cunningham, W. & Jefferies, R. (2000, July/August). Strengthening the case for pair programming. *IEEE Software*, pp. 19-25.

Wood, J. & Silver, D. (1989). *Joint application design: How to design quality systems in 40% less time.* New York: Wiley

Yourdon, E. (2000). Success in e-projects. *Computerworld, 34,* (34).

Zikmund, W.G. (2003). *Business research methods*. Mason: South Western.

Biographies

Glen Van Der Vyver is a senior lecturer in Information Systems at the University of Southern Queensland. He worked at the "coalface," in Human Resources and IT (mostly), for around fifteen years before becoming an academic. He leads the database "stream" and teaches the core database courses. Glen has a varied academic background and research interests, ranging from core interests relating to IT careers, teaching IT, E-Business and risk perception to more personal interests in literature, art history and film.

Michael Lane is a lecturer in the Division of Information Systems, Faculty of Business, University of Southern Queensland. He holds an Honours in Information Technology from the same University and is currently enrolled in a PhD with a focus in E-Commerce Development. He has published widely in Information Systems and Electronic Commerce but varied interests in research and teaching. He has run a postgraduate course on E-Business Strategy at the Australian Graduate School of Business at USQ for the last four years.