Journal of Information Technology Education: Innovations in Practice **Volume 7, 2008**

From Requirements to Code: Issues and Learning in IS Students' Systems Development Projects

Elsje Scott
University of Cape Town, Cape Town, South Africa

elsje.scott@uct.ac.za

Executive Summary

The Computing Curricula (2005) place Information Systems (IS) at the intersection of exact sciences (e.g. General Systems Theory), technology (e.g. Computer Science), and behavioral sciences (e.g. Sociology). This presents particular challenges for teaching and learning, as future IS professionals need to be equipped with a wide range of analytical and critical thinking skills that will enable them to solve business problems. In addition, they require technical, strong interpersonal communication, and team skills to contribute to the successful delivery of software products.

At the University of Cape Town (UCT) the capstone course of the IS undergraduate curriculum is structured around three main areas: Project Management; People Management; and Implementation. The theoretical parts of this course introduce the student to important aspects of managing projects and people in the Information Communication and Technology (ICT) Project environment. The practical part comprises a group systems development project, which forms a core part of the course and requires students to apply theoretical skills in a real-world context. Although the impact of the issues relating to soft skills on student learning is neither underestimated nor ignored in the course, this paper mainly focuses on the technical issues that are experienced during the life of the projects.

Students generally experience difficulty in the areas of problem-solving, coding and testing, all of which are required for successful systems development. IS students are often less technically oriented than their counterparts in the other computing disciplines and their courses involve less technical content. As a result, they may be inadequately prepared for the technical demands of the project. IS professionals must be able to interact with business experts and apply problem-solving skills in developing possible solutions. It is thus reasonable to argue that the completion of a full life cycle of a project provide IS students with invaluable experience in testing the effectiveness of their proposed solution.

A reflective approach has been applied to the course design, resulting in the development of a framework to sufficiently address the issues of problem-solving, coding, and testing through an action learning cycle. This approach has proved to lead to improved solutions and to encourage deep learning. It also shows how teaching practices are shaped by looking back reflexively of student learning and the facilitating environments.

This paper describes how the course has evolved through four phases, culminating in an approach that guides students

Editor: Linda Knight

Note: An earlier version of this paper appeared in the Conference Proceedings of the Computer Science & Information Technology Conference 2007 (CSITEd2007), University of Technology, Mauritius.

in transcending from the basic level of following, through detachment towards fluency. In this fourth and current phase two pilot projects have been included to develop a framework or pattern that will provide students with a sound basis for developing their own software system in the second part of the course. This framework uses a methodology to structure large software-intensive systems into modular components that can be developed and maintained independently. It follows a recursive process where students first develop an independent component, and then add a dependent component to form a larger but again independent component. The repetitive implementing of the framework through three iterations (two pilot projects and final group project) promotes the transfer of skills and problem–solving techniques to similar situations and problems and aids students to overcome their fears and anxieties when faced with problem situations.

Several relevant studies have been undertaken over the years to encourage and support the critical reflective approach, deep learning, and improved solutions. A longitudinal study was conducted in 2007 to determine the readiness of project teams to start the building phase by the end of the first semester of the group project. This study will be extended to evaluate the impact of the improvements made during the fourth phase of the project, with specific reference to the issues identified in this paper.

Keywords: Information Systems, Systems development group project, Action learning, Deep learning, Problem solving, Object-Oriented paradigm/environment.

Introduction

Information Systems (IS) is one of five computing disciplines identified by Computing Curricula (2005), the others being Computer Engineering, Computer Science, Information Technology, and Software Engineering. However, because IS lies at the intersection of exact sciences (e.g. General Systems Theory), technology (e.g. Computer Science) and behavioral sciences (e.g. Sociology), it presents particular challenges for teaching and learning. According to IS2002, An Update of the Information Systems Model Curriculum (Gorgone et al., 2002), the IS discipline depends on three key attributes:

- A broad business and real world perspective.
- Strong analytical and critical thinking skills.
- Strong interpersonal communication and team skills.

A key role of the IS professional is to determine the requirements for an organization's information systems and to play an active role in the specification, design and implementation thereof (Computing Curricula, 2005). In doing so, IS professionals are faced with the challenge of translating a variety of business processes into information technology solutions that will meet the needs of the organization both efficiently and effectively.

For decades, the IS industry has experienced problems of software that is not delivered on time or which is outdated before it can be implemented, and systems where requirements have not been met or which are completely unusable (Parnas, 2006). It is thus essential for IS education to provide a 'base foundation' of skills and knowledge that will equip future IS professionals for the effective delivery of information systems (Phukan, 2001). Requirements that are misinterpreted or not fully understood will lead to flawed design resulting in an unsuccessful product. It is thus crucial that IS students are given the opportunity during their undergraduate degree program to test their understanding of requirements and their design skills by implementing them though the development of a software product. In addition, the rapidly changing field of computing places stringent demands on IT/IS educators to continually revise and change programs and curricula in an attempt to better equip students for the marketplace (Dawson & Newman, 2002; Kussmaul 2000; Noll & Wilkens, 2002; Tuttle, 2000).

The Computing Curricula (2004, 2005) depict IS as an applied discipline, primarily concerned with the relationship between information systems and organizations. The IS professional is, amongst other roles, concerned with the tailoring of database applications, the development, deployment and configuration of systems to suit the needs of organizations, and the training of users. To successfully prepare students for these roles they have to understand the full systems development lifecycle, irrespective of the specific methodology that may be used to develop a software system. This involves the inception phase of the project, the analysis and design phases, the building and the implementation phases. The process can follow the traditional model or involve the iterative building of consecutive subsystems. Whichever method is followed, testing must be included in the different phases to ensure the quality of the final product.

A systems development group project is the main deliverable of the capstone course Project Management: Theory and Implementation, offered to Information Systems majors at the University of Cape Town (UCT), South Africa. This third year course, and more specifically the project as the main deliverable of the course, has been subjected to an ongoing refinement process over the past eight years. It has been shaped according to guidelines provided by computing curricula (Computing Curricula, 2004, 2005; Gorgone et al., 2002;) and influenced by various teaching and assessment theories including that of Cockburn (2002), which suggests a framework based on the evolutionary path that developers tend to follow.

In its current state the course is structured around three main areas: Project Management, People Management, and Implementation. The theoretical parts of this course introduce the student to important aspects of managing projects and people in the Information Communication and Technology (ICT) Project environment. The practical part of the course involves the application and implementation of these concepts while following the full life cycle of a project using a team-based IS project in a real-life setting. This paper mainly focuses on the technical issues that are experienced during the life of the projects and not those relating to soft skills. The impact of the issues relating to soft skills on student learning however is neither underestimated nor ignored in the course. The Project Management section includes a comprehensive overview of the nine core components of the Project Management Body of Knowledge (PMBOK), while innovative ways are used to design and guide project teams through the lifecycle of the group project.

Several problems are commonly encountered during the systems development project lifecycle, although the degree of severity may vary depending on the specific methodology being used.

- Students struggle with abstraction and problem solving. Because of their own lack of business experience, students often have difficulty in understanding business requirements, and the abstraction involved when translating requirements into code presents a considerable challenge.
- Students experience difficulties with fundamental coding principles. IS students are often less technically oriented than their counterparts in the other computing disciplines and their courses involve less technical content. As a result they may be inadequately prepared for the technical demands of the project.
- Students fail to test the systems they develop efficiently. The validation and verification of the implemented processes during the development cycle is ineffective, insufficient, and is not thoroughly executed.

This paper examines student learning within the context of the systems development group project, with particular focus on the issues identified above. The paper further explores the literature to obtain a better understanding of the elements of different kinds of learning and the role of a teacher in facilitating learners' development. It then reports on the transition phases of the project

course as it has developed over the past eight years, and describes the attempts made to effectively address these issues and deliver competent students for professional practice.

Issues in the Programming Environment

The competence of an IS professional is reflected in the effectiveness with which analysis and design specifications can be translated into code. "Modern software practices call for the active involvement of business people in the software process" (Roussev, 2003, p. 349). This means that IS professionals must be able to interact with business experts and apply problem-solving skills in developing possible solutions. It is thus equally important that programming form a core component of an IS course to ensure that students acquire the necessary programming skills to be able to design a system that can be translated into a rigorous software solution.

Problem Solving

Students ranked logical thinking and problem-solving skills as the most important ability for learning programming in a multi-national, multi-institutional survey done by Simon et al. (2006). In a similar industry survey, problem-solving ranked sixth in the list of knowledge, skills and abilities needed by an entry-level computer programmer (Simon et al., 2006).

Keller and Concannon (1998) define problem-solving as a vital but basic life skill that entails the solving of new problems in terms of analogies of previously learned procedures. Pedagogical (teacher-centered) or methodological (learner-centered) strategies can be implemented to overcome barriers that might prevent students from solving problems effectively. The UCT group project implements a methodological strategy using a step by step method to assist students in solving their business problem. Table 1 below portrays the analogy of the IDEAL heuristic method of Bransford and Stein as listed by Keller and Concannan (1998) and the phases in the group project. This method represents the five steps usually contained in many solution strategies.

Table 1: Analogy of the IDEAL heuristic and the phases of the group project

Steps of IDEAL heuristic	Phases of the group project
Identify the problem	Finding and understanding the business problem
Define and represent the problem	Analysis phase of defining the user requirements
Explore possible solution strategies	Systems design and technical specifications
Act on the strategies	Building phase
Look back and evaluate	Final testing, validation and presentations, lessons learned

General Coding Problems

It is well known that students experience significant difficulties in learning to program and in mastering fundamental coding concepts (Bergen & Reilly, 2005; Simon et al., 2006). These issues are often more prominent in the IS discipline, since in-depth coding courses, exposure to low level programming languages and rigorous algorithmic approaches do not normally form part of the typical undergraduate IS program. This means that IS students frequently lack the knowledge of basic principles that could provide a foundation for better understanding of coding in general. In addition they struggle to understand the abstractions that are required in the Object-Oriented (OO) environment. For example, they have difficulty in defining a class of objects that includes the attributes and behaviors of objects of this kind on the one hand, and using an object of this kind on the other hand. Students struggle to comprehend that:

- an object must first be instantiated (created) before it can be used.
- its state can change overtime as it receives different values for its attributes, and
- it can perform different actions, depending on the messages it receives from its environment.

Previous experience on this project has shown that student find it difficult to write OO code for these different perspectives.

Testing

Extensive testing is a necessary and crucial step in the systems development process. Testing, complemented by formal inspections in the early stages of a project are essential activities towards ensuring that a system is working correctly. It can also aid to demonstrate that the developers have understood and met customers' requirements (Tayntor, 1998). An empirical research study done at UCT in 2004 indicated that the standard of software testing is lower in South African (SA) companies than in non-SA companies. The study, however, concluded that SA software developers, who have adopted the OO development methodology, have also implemented the formal testing practices associated with it (Scott, Katovsky, Burdzik, & Elley, 2004). Students who are exposed to testing practices during their undergraduate studies will develop a better understanding of professional practice and gain corresponding analytical and critical thinking skills.

Teaching and Learning

Teaching practices have evolved from the approach where a teacher is in total control of the content being transferred (transfer theory), or is responsible for shaping the students' viewpoints (shaping theories), to more developed theories where the driving force for learning and growth is internal to the student (growing theories) (Fox, 1983). Experiential learning and projects are seen as teaching strategies derived from developed theories. Here the emphasis is on the activities of the student and the influence of these activities on his or her learning (Fox, 1983).

These activities encourage students to assume increased responsibility for their own learning. When this happens it is essential for the instructor to study the students' learning process so as to gain an insight into their understanding and application of the concepts and methods encountered in the field of study (Ramsden, 2003). Such insight will enable the correction of deficiencies or misconceptions and improve students' abilities to solve real world problems.

Students have to be supported through what Dreyfus and Dreyfus (1986/1988) called the five stages of adult learning as being: Novice, Advanced Beginner, Competence, Proficiency, and Expertise. This transcendence defines an evolving education that moves from unconscious incompetence to conscious incompetence through conscious competence until unconscious competency is finally reached. A fundamental task of teachers is thus to encourage the engagement of students in learning activities as this will heighten students' enjoyment and achievement levels, resulting in deep learning (Biggs, 1993).

In many subject areas projects are used as vehicles to engage students in an Action Learning Cycle, a cycle that promotes continuous planning, reflection, observation, and action amongst participants (Bunning, 1997, as cited by Machanick, 2005). Often these projects form an integral part of a capstone course, designed to assess the command, analysis, and synthesis of knowledge and skills in a student-centered manner (Moore, 2005). Moore (2005) reports that a capstone course serves as "an instrument of evaluation in all three modalities of learning": cognitive, affective and psychomotor.

The Systems Development Group Project

The third year systems development group project (subsequently referred to as the group project) at the Department of Information Systems, UCT, has evolved to its current state through four clearly identifiable phases. Mitigation strategies and a reflective approach have been adopted on an ongoing basis to identify and address issues as they manifested over time in the group projects.

Several important elements have, however, always been present throughout the existence of the group project. The group project has a life span of approximately 7 months, from mid February to mid September of each year. A real world perspective and a broad business background have always formed a core focus of the project. Student teams of four or five members each are required to find a sponsor (client) in industry to provide them with a suitable business problem. The sponsors are available for meetings and queries so as to provide guidance to the teams regarding user requirements and business processes, but no monetary assistance is provided.

Although teams manage their own work, faculty members, acting as project managers, monitor their progress and help to ensure that the scope of the project is in accordance with the project specifications.

Faculty members are also involved in the final assessment of the projects, which takes the form of a live presentation by student teams of each of the individual projects. This is a formal occasion requiring the participation of all the team members.

At the completion of the group project, many students are of the opinion that the group project has exposed them to many of the challenges that they would experience in their employment. They feel that it provided them with good "practice runs" before they had to work on actual projects for their respective employers. The following quote from an alumnus reflects this viewpoint: "The structure of the project that I have been working on was very similar to that of the university projects and I felt far more confident doing this project knowing that I had already done two similar projects and encountered and overcome many of the issues associated with IT projects."

The following sections tell a story of the evolution of teaching in the group project. It is an account of several iterations of reflexive learning of student learning, influenced by the education process. The account relates how ongoing transcendence occurred from being unconsciously incompetent to being consciously competent.

Phase 1 (2000)

In 2000 the group project constituted a major component of a full year project management course. Support was provided to students in the form of regular lectures and tutorials, as well as project-specific functionality guidelines. Although teams used some Unified Modeling Language (UML) artifacts during the analysis and design phases, few other structures existed to guide students in developing problem solving abilities and testing techniques. In addition, students were only exposed to coding in the introductory coding course in their first year and then again in the project in their third year. As a result, many team members lacked technical and coding skills.

The projects were assessed at the end of the life cycle, using standardized mark sheets for the final evaluation. Although the functionality of the systems was evaluated and tested thoroughly during this assessment, the systems were taken at "face value" and only the executable versions of the systems were examined. Since the coding was not evaluated, it was possible that student teams could obtain good marks without adhering to good programming principles or rigid standards. Frequently the students' inability to solve business problems effectively meant that the design models they developed were not sufficiently rigorous to form the basis for successful coding. In these cases student teams would reverse engineer the design model from the working sys-

tem to correlate with the code. In other cases, projects did not exhibit multi-tier architectures; and often the documentation that was submitted was not in line with the actual software product.

The general lack of crucial technical competencies caused severe imbalances between project teams, with many students failing to experience the true meaning of action learning. This is in line with the observation by Machanick (2005) that students often apply possible solutions to programming and team problems randomly instead of reflecting on the outcomes of previous solutions before the cycle of planning, action, and reflection is repeated. On the whole it became apparent that students often performed activities without a clear appreciation and understanding of the processes involved. They were clearly in need of careful guidance to help them reach the required level of maturity in their understanding, problem solving, and other skills. Because of these concerns, the faculty felt it necessary to re-think the project course and increase the scaffolding provided within the learning environment.

Phase 2 (2001-2002)

In an attempt to address the limitations of the previous phase more effectively, the group project was instituted as a new and completely separate course offered over three terms, excluding the theoretical project management sections. A generic theme was identified for each year's projects, with a distinct deliverable structure. Within this framework, students were expected to identify a business problem and develop an appropriate solution. The project was broken down into clear interim phases: project definition, system analysis, design, building, and testing, each of which incorporated systems development deliverables, project management deliverables, and quality control procedures. These deliverables guided students in the application of previous knowledge (theoretical and practical) and in the acquisition of new specialized skills needed to elicit user requirements, solve their specific business problem, understand scope, and complete the analysis, design, and building phases. These were evaluated by the project manager, and the feedback was used to improve the interim deliverables, culminating in a milestone deliverable for each phase of the project. The use of UML artifacts were extended and used more effectively during the analysis and the design phases in order to avoid the reverse engineering of the models at the end of the project. These processes reflect the notion that critical thinking, critical being, and transformation underpin the discourse of higher education (Doyle, n.d.). The project culminated in a final deliverable, being the complete shrink wrapped product consisting of all the documentation and the software system. In addition, an Expo event that showcased all projects to industry, learners from nearby schools, and the wider public was initiated in 2002.

Additional support was provided through seminars on relevant technical topics. In an attempt to address the deficiency in technical skills apparent in the previous phase, specific technical topics were identified. Student teams comprising representatives from different project teams were each given a topic to research, prepare, and then present to other members of the class during a seminar session. Technical concepts had to be explained and demonstrated by accompanying documents and software programs; each team had sessions with the course convener prior to their presentation to ensure that the material presented was of the required standard. An additional benefit of these seminars was that the students who presented a particular technical topic would subsequently take their new skills back to their own project teams.

In keeping with the view that assessment is necessary for teaching to enhance and support the learning process (Shepard, 2000), a more comprehensive assessment strategy was also introduced during this phase and is described in more detail in Scott and Van der Merwe (2003). This strategy was developed around the principles of comprehensiveness, coherence, and continuity advocated by Pellegrino, Chudowsky, and Glaser (2001), and included elements of dynamic assessment, assessment of prior knowledge, the use of feedback, teaching for transfer, student self-assessment, and the evaluation of teaching, as proposed by Shepard (2000).

A variety of instruments, such as checklists, questionnaires, tests and examinations, mark sheets, and scoring rubrics, were designed in support of this assessment strategy. These were specifically intended to provide students with significant feedback and to encourage what Entwistle (2001) describes as a deep approach to learning. Scoring rubrics formed the backbone of the assessment instruments used in this phase and were found to be very effective mechanisms in eliminating bias and conveying to students the standards against which they would be measured. (Rubrics can be defined as rating and scoring guides with predetermined criteria; Metler (2001) provides useful detail on this topic.) Ongoing formative assessment provided rich feedback on the interim deliverables, allowing students to continuously and dynamically improve these deliverables towards the milestone deliverable.

The introduction of this comprehensive assessment strategy was very successful in providing structure and guidance to students that saw them become part of the learning process, and transcend and improve their problem solving skills. Similar success however was not achieved through the implementation of the technical topics. Not all students benefitted equally from the material presented and the seminars did not produce the student commitment and deep learning that would usually be expected from a more hands-on approach. Although the topics provided students with material and directions for tackling certain tasks when developing their software systems, they did not sufficiently address the fundamental coding issues and deficiencies in coding skills that existed. Further enhancements to the project course were clearly needed to improve the development of problem solving skills and coding abilities.

Phase 3 (2003-2006)

During this phase the group project was recombined with the project management and people management modules to form a full year course. This was done in order to establish a discourse that would encompass the full range of characteristics of the IS discipline rather than emphasizing a single dimension. Within the course, the group project itself was characterized by a strong technical focus.

A generic theme was still identified for the group project in this phase, but the solution had to be a comprehensive management system, incorporating a back-end system and a web component. More emphasis was placed on the efficient interpretation and capturing of the business problem. Package and activity diagrams were added to the suite of UML artifacts to respectively assist students to obtain an overview of the system and to enhance their understanding of the main business processes. The solution had to be credible and convincing, developed using sound business rules and processes. The assessment approach was therefore further refined to evaluate how well the solution reflected the essence and identity of the business. The recognition of individual contributions to group performance in these projects could not be ignored, and it also became necessary to include peer assessment as part of the assessment strategy. A rubric was further developed for the marking of the code of the final product and included as part of the summative assessment.

The designated technology platforms used during this phase were Visual Studio.NET 2003 and the SQL Server 2003 database engine. (This platform has since changed to the 2005 versions of these products.) The current technologies implemented in the group project are listed in Table 2 in the Appendix.

Students were expected to implement object-oriented design and programming principles within an n-tier distributed environment. Unfortunately novice developers struggle with even simple 2-tier applications and are likely to be completely out of their depth in n-tier environments (Lhotka, 2006). Because of this, it is crucial to establish a well thought through framework and pattern

that can be applied repeatedly when developing highly scalable and maintainable object-oriented business applications.

The technical topics implemented in the previous phase had not prepared students adequately for the complex task of translating requirements into code. An alternative approach introduced in phase three was to walk students through a similar "pilot" system, using a step by step approach that was developed over a number of workshops. This provided students with coding skeletons and a framework for developing integrated object-oriented n-tiered systems that they could utilize again and again for subsequent software development. "In the software world, the easiest way to reduce overhead is to increase reuse, and the best way to get reuse out of an architecture (both design and coding) is to codify it into a framework" (Lhotka, 2006, p. 33).

While working through this pilot system, students were constantly confronted with aspects of problem solving and had to practice innovative ways of translating requirements into code. They conquered many basic coding issues that were previously problematic, and became more comfortable with the advanced concepts of the OO paradigm such as encapsulation, inheritance, and polymorphism. They also learnt how to identify objects, to create integrated environments that would handle interdependencies between objects, and to deal with events.

By providing teams with a framework to guide their own system development efforts, the pilot system definitely improved the quality of the final products. Although students did not initially realize the value of the framework as provided in the workshops, overwhelmingly positive feedback was received in the form of comments accompanying the student course evaluations after the completion of the project. This series of workshops was clearly successful in guiding students through at least the first four (Novice, Advanced Beginning, Competence, and Proficiency) of the seven stages of skills development identified by Dreyfus (2001). However, the "hand holding" provided by the step by step approach of the pilot system may well have stifled innovation and creativity among students, so that few transcended to the subsequent stages of expertise, mastery, and practical wisdom (Dreyfus, 2001).

According to Biggs (1993) the strategy of the "deep approach" is to maximize understanding by ensuring that the student's learning is based on his or her interest in the subject matter or the tasks involved. Despite attending the workshops, some students did not come to a full understanding of the concepts that were being demonstrated and did not achieve a sense of mastery. This issue initiated the fourth and current phase in the evolvement of the capstone course.

Phase 4 (2007)

In the current phase, the aim is to develop and reinforce students' understanding of the advanced OO concepts, coding patterns, and framework provided through the pilot system, by requiring that they simultaneously develop a system based on a second case study with the same generic theme as the current project. This provides a context within which students learn new skills through the pilot project, apply these skills in the second case study, and should thus be equipped to use them more effectively in their independent group project. A workshop focusing on the second case study is held after every second workshop of the pilot system, with the primary objective of mastering the concepts learned in the previous two workshops of the pilot system. Almost no scaffolding and hand holding is provided during the workshops for the second case study, and students are challenged to demonstrate their own understanding and expertise. This approach follows Cockburn's (2002) three stages of development: namely, following, detaching, and fluency. Class tests based on mini case studies are used to further accelerate students' understanding of core coding concepts.

A secondary objective of this phase is to achieve a greater degree of deep learning amongst students and by their own reflexive approach enhance their parsing patterns, improving their ability

to think and reason more formally about their programs. The parsing pattern is what Cockburn (2002) identifies as the interpretation filter. The hope is that this additional step in the learning cycle will enhance the preparedness of the student teams and provide them with sufficient enthusiasm and inner drive to embark on the building phase of their individual projects.

Another aspect of systems development that is receiving attention in the current phase is the issue of quality. In addition to formal inspections early in the development cycle, testing constitutes an important mechanism to verify the effectiveness of the solution (Tayntor, 1998). A comprehensive testing plan forms part of the requirements for the group project; dedicated workshops and lectures on testing are conducted during the course, and student teams are encouraged to do comprehensive testing of their systems. Nevertheless, many group projects are not of a high enough standard to be implemented immediately in a business environment as is, at the hand-in date. This suggests that testing is not being performed to sufficiently rigorous standards and does not cover the wide spectrum of tests usually performed by industry. To address this problem, the evaluation of the group project has recently been extended by adding a mock implementation of the system to the code walkthrough and formal presentation that were previously included in the final summative assessment.

Conclusion

Teachers face a challenge in adopting the practice of critical reflection: a conscious awareness of the "what", "why" and "when" of student learning where "Practice is informed by theory that has been shaped by practice" (Walkington, Christensen, & Kock, 2001, p 344). However, the development of this competency will facilitate the continuous enhancement of teaching and student learning. The study shows how teaching practices are shaped by looking back reflexively of student learning and the facilitating environments.

The group project deliverable at UCT contains numerous ongoing processes and activities to ensure that teaching is tightly correlated with learning and assessment which go beyond just knowing and facilitates deep learning. The underlying premise of the two pilot projects included in phase 4 is to develop a framework or pattern that will provide students with a sound basis for developing their own software system in the second part of the course. This framework uses a methodology to structure large software-intensive systems into modular components that can be developed and maintained independently. It follows a recursive process where students first develop an independent component, and then add a dependent component to form a larger but again independent component.

The repetitive implementing of the framework through three iterations promotes the transfer of skills and problem–solving techniques to similar situations and problems and aids students to overcome their fears and anxieties when faced with problem situations. During the first iteration (following phase) students mainly follow a step by step process to cope with the design and development of dependent and independent components. In the second iteration (detachment phase) they implement the methodology without the assistance of scaffolding, while the third iteration (fluency phase) calls for innovation and creativity in building a working system and customizing it to meet industry requirements. These cycles encourage an action learning process in which students transcend from the following stage through the detaching stage towards becoming fluent IS practitioners.

Previous research by Brown and Pearce (2006) on factors affecting the success of UCT group projects was based on interviews with fourth year IS students and may have included an element of bias as their learning experiences had already been influenced by six months of participation in their fourth year level course. A longitudinal study is currently underway at UCT to determine the readiness of project teams to start the building phase by the end of the first semester of the

group project. This study will be extended into the second semester of 2007 in order to evaluate the impact of the improvements made during the fourth phase of the project, with specific reference to the issues identified in this paper.

Acknowledgement

I am grateful to Jane Nash for editing the first version of this paper. Her useful comments and advice were invaluable in enhancing this paper.

References

Bergen, S., & Reilly, R. (2005). Programming: Factors that influence success. *Proceeding of the Thirty Fifth SIGCSE Technical Symposium on Computer Science Education,* 411-415. St. Louis: ACM.

Biggs, J. (1993). What do inventories of students' learning processes really measure? A theoretic review and clarification. *British Journal for Educational Psychology, 63*, 3-19.

Brown, R., & Pearce, J. (2006). *Does the use of software development standards have an impact on software development projects: An investigation of software success and quality factors?* An empirical research paper presented to the Department of Information System at the University of Cape Town, South Africa.

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.

Computing Curricula. (2004). *Computing curricula: Overview report including a guide to undergraduate degree programs in computing* [November, 22 Draft]. Retrieved April, 7, 2007 from http://www.acm.org/education/Overview_Draft_11-22-04.pdf

Computing Curricula. (2005). *Computing curricula. The overview report including the guide to undergraduate degree programs in computing* [April, 11 Draft]. Retrieved April, 7, 2007 from http://www.acm.org/education/Draft_5-23-051.pdf

Dawson, R. J., & Newman, I. A. (2002). Empowerment in IT education. *Journal of Information Technology Education*, *1*(2), 125 – 142. Retrieved from http://jite.org/documents/Vol1/v1n2p125-142.pdf

Doyle, M. (n.d.). *A reflexive critique of learner managed learning*. Retrieved April 13, 2007, from http://www.leeds.ac.uk/educol/documents/00002420.htm

Dreyfus, H. (2001). *On the Internet*. London: Routledge.

Dreyfus, H., & Dreyfus, S. (1986/1988). *Mind over machine: The power of human intuition and expertise in the era of the computer*. New York: Free Press.

Entwistle, N. (2001). Promoting deep learning through teaching and assessment. In L. Suskie (Ed.), *Assessment to promote deep learning: Insights from AAHE's 2000 and 1999 assessment conferences,* 9 – 20. Washington, D.C.: American Association for Higher Education.

Fox, D. (1983). Personal theories of teaching. *Studies in Higher Education, 8*(2), 151-163.

Gorgone, J., Davis, G., Valacich, J., Topi, H., Feinstein, D., & Longenecker, H. (2002). *IS 2002: Model curriculum and guidelines for undergraduate programs in information systems*. Retrieved December 3, 2003, from http://www.acm.org/education/is2002.pdf

Keller, R., & Concannon, T. (1998). Teaching problem-solving skills. Presented at a *Workshop for the Centre of Teaching and Learning*. Retrieved April 10, 2007 from http://ctl.unc.edu/fcy20.html

Kussmaul, C. (2000). A team project course emphasizing software entrepreneurship, *Proceedings of the Fifth Annual CCSC Northeastern Conference on the Journal of Computing in Small Colleges*, 313 – 321.

Lhotka, R. (2006). *Expert VB business objects*. Berkeley, California: Apress.

Machanick, P. (2005). Peer assessment for action learning of data structures and algorithms. *Conferences in Research and Practice in Information Technology*, *43*, 73-82, Newcastle: Australian Computer Society.

Metler, C.A. (2001). Designing scoring rubrics for your classroom. *Practical Assessment, Research & Evaluation*, *7*(25). Retrieved April 20, 2003 from http://edresearch.org/pare/getvn.asp?v=7&n=25

Moore, R. C. (2006). Direct measures: The capstone course. In W. G. Christ (Ed.), *Assessing media education: A resource for educators and administrators* (Chapter 21, 439-459). Hillsdale, NJ: Erlbaum.

Noll, L. C., & Wilkens, M. (2002). Critical skills for IS professionals: A model for curriculum development. *Journal of Information Technology (JITE), 1*(3), 143-154. Retrieved from http://jite.org/documents/Vol1/v1n3p143-154.pdf

Parnas, D. (2006). Agile methods and GSD: The wrong solution to an old but real problem. *Communications of the ACM*, *49*(10), 29.

Pellegrino, J. W., Chudowsky, N., & Glaser, R. (Eds). (2001). *Knowing what students know: The Science and Design of Educational Assessment*. Washington, DC: The National Academic Press.

Phukan, S. (2001). Changing education to meet the needs of future informing science clients: Suggestions for new curriculum frameworks. *Proceeding of the Informing Science & IT Education Conference (IS2001),* Krakow, Poland, pp. 409-414. Retrieved from http://proceedings.informingscience.org/IS2001Proceedings/pdf/PhukanEBKchang.pdf

Ramsden, P. (2003). *Learning to teach in higher education*. London; New York: RoutledgeFalmer.

Roussev, B. (2003). Teaching introduction to programming as part of the IS component of the business curriculum. *Journal of Information Technology Education, 2*, 349-356. Retrieved March 12, 2007 from http://www.jite.org/documents/Vol2/v2p349-356-43.pdf

Scott, E. C., Katovsky, B. Burdzik, J., & Elley, T. (2004). The impact of the adoption of object-oriented testing practices – A South African perspective. *Proceedings of the Joint International Conference on Informatics and Research on Women in ICT (RWICT) 2004, Kuala Lumpur, Malaysia*, 423-442.

Scott, E. C. & Van Der Merwe, N. (2003). Using multiple assessment approaches to enhance objectivity and student learning. *A special conference edition of the Electronic Journal of Information Systems Evaluation (EJISE), 6*(2), 182 – 186.

Shepard, L. A. (2000). The role of assessment in a learning culture. *Educational Researcher, 29*(7), 4-14.

Simon, Fincher, S. A., Robins, A., Baker, B., Box, I., Cutts, Q., et al. (2006). Predictors of success in a first programming course. *Proceeding of the Eighth Australasian Computing Education Conference (ACE2006).*

Tayntor, C. B. (1998). Software testing basics and guidelines. *Information Management: Strategy, Systems, and Technologies*. Auerbach Publications.

Tuttle, S. M. (2000). A capstone course for a computer systems major. *Proceeding of the Thirty First SIGCSE Technical Symposium on Computer Science Education*, 265-269.

Walkington, J., Christensen, H. P., & Kock, H. (2001). Developing critical reflection as a part of teaching training and teaching practice. *European Journal of Engineering Education, 26*(4), 344-350.

Appendix

Table 2: A summary of the common technologies used in the group project	
TECHNOLOGY	**IMPLEMENTATION**
Microsoft Project Server (Enterprise Edition) 2007	The technology platform for workgroup environments
SharePoint Team Services/Time Sheets	Enable team collaboration
Microsoft Project 2007	The Gantt Chart is utilized to establish and maintain the project planning
Microsoft Office Visio 2007	Used as a tool to produce Unified Modeling Language (UML) artifacts in the analysis & design phases of the project
Microsoft Office Word 2007	Utilized to enable interoperability with Visual Studio in the group projects and to produce the documentation
Microsoft Office Excel 2007	Utilized to enable interoperability with Visual Studio in the group projects
Visual Basic.NET 2005	Used as the main programming platform
ASP.NET 2.0	Implemented to create the Web development sections of the projects
Microsoft SQL Server 2005 / Microsoft SQL Express	Most databases used in the group projects are created in SQL or SQL Express (In some cases students may also use Microsoft Access 2007 or MySQL)
Crystal Reports	Used for efficient reporting capabilities in the applications
Visual Source Safe	Used to enable versioning of the software

Biography

Elsje Scott is a Senior Lecturer at the Department of Information Systems, University of Cape Town. Her main research interest is systems development group projects which include knowledge areas like project management, people management and software engineering. Specific focus areas are software testing, object-oriented programming concepts, general issues concerning the development of efficient computer systems in Information Systems, assessment strategies and software standards.