

Teaching Software Development to Information Systems Majors Using an Action Research Process

Mr Kosheek Sewchurran
University of Cape Town
ksewchurran@commerce.uct.ac.za

Mrs Elsje Scott
University of Cape Town
Escott@commerce.uct.ac.za

Mr Mike Eccles
University of Cape Town
meccles@commerce.uct.ac.za

ABSTRACT

Information Systems(IS) students are often inundated with conflicting, voluminous advice about development practices. Their confusion is occasionally further exacerbated when they attempt to put the theory into practice and find that practice is not well correlated with the discourse taught in an Information Systems major (Cockburn(2002); Burns and Klashner(2005)). This experience can often result in significant trauma for the student and the student begins to doubt the education process. As IS academics we design the learning process and prepare students for careers in the IT industry. A framework presented by Cockburn (2002), which describes the evolutionary path developers tend to follow, as well as other approaches researchers have found beneficial, are presented as a framework of ideas which are being used to shape the curriculum design at the University of Cape Town (UCT). This paper discusses the application of this framework and the results obtained thus far.

Keywords

Systems Development, IS Curricula, Agile, Interpretive Information Systems Research, Action Research

INTRODUCTION

At the University of Cape Town (UCT) the development thread of the IS major is characterised by the following core modules: Systems Analysis and Design, Information Systems Project Management and Component based development. These modules are taught over a four year period. The detailed IS development curriculum is as follows:

During the first year students are introduced to the underlying concepts underpinning systems development and the Software Development Life Cycle (SDLC). Students learn how to programme in VB.net, but their exposure to systems analysis and design is limited to a few theoretical concepts.

Half of the second year syllabus is comprises object oriented systems analysis and design (OOAD). The OOAD syllabus includes the following topics:

1. Systems theory and the role of the systems analyst
2. Developing a Business Case
3. User requirements elicitation
4. Developing the Analysis Model including use case, domain class, activity, interaction and state machine diagrams
5. Developing the detailed Design Model including systems architecture, interface, class and database design
6. Additional programming to provide students with the skills to build systems to a multi-tiered architecture.
7. Constructing and implementing of a partial solution including testing and rollout strategies.
8. Detailed coverage of systems development methodologies (traditional, iterative and agile)

The second year discourse is designed to impart the core concepts, tools and techniques. Each week discusses a core concept along with the necessary tools and techniques relevant to the core concept. A typical week will include a couple of lectures and a tutorial session to emphasise practice of the theory. The tutorial session allows students to be hand held by the lecturer and tutors. Assistance is provided while the students apply themselves. The attempts are marked and returned with feedback by the following week. A common theme holds all the tutorial sessions together and class examples together. In the second semester the students working in teams of two model and develop the artefacts mentioned above and build a part of the system for presentation and evaluation.

The second year teaching is carefully collated to maximise learning. Firstly, the problem domain is not complex and is drawn from areas familiar to most students, staff and tutors. Due to a shared understanding of the problem domain, feedback to students is rich and common problems and concerns are discussed and

resolved in lectures. Secondly, the scope of the application is carefully managed. The business case covers the entire system under review, but the analysis is scoped to a single subsystem with detailed design being limited to a few use cases. Finally the students are all participating on a common development project where collaboration and advice on each life cycle artefact is easily accessible.

At 3rd level no teaching in analysis and design is explicitly performed although students are re-introduced to core concepts and techniques with refresher lectures, documentation guidelines and templates. The core discourse at 3rd level consists of: Project management theory; Client Server and Internet applications architecture and programming using VB.net™; Soft skills which include group interaction and leadership; and finally a group development project.

The group development project is also referred to as the capstone course, and requires students to participate in a non-trivial development project of a particular theme. The group development project requires the application of all their acquired skills. The most recent theme has been inventory management. Students are required to work in groups of 5 or 6 focussing on addressing a real world problem directly related to the theme. The development project process is supported by members of the academic staff who act as project mentors and provide guidance and ensure milestone adherence. The teaching environment in third year is carefully scaffolded because of the extent of the planned discourse. Apart from formal theory lectures, students participate in weekly workshops and practical sessions to apply project management and advanced programming concepts.

The development project is the major deliverable in third year and each team is required to find a suitable application in a Cape Town organisation, arrange sponsor approval and develop the system over a six month period. The project is of considerable scope and complexity, and although the department has had tremendous success with putting some students through the above bearable amount of pressure, and see them transcend, some students become traumatised by the process. The trauma seems to result from one or more of the following:

1. The realisation that modelling and following rigorous process does not necessarily lead to success. This could be classified as a “painting by numbers” syndrome. Students often perform an activity without a clear appreciation of the role and value the activity provides in the overall development process. In a structured “ring in the nose” teaching environment these phenomena is not that easy to identify. In a third year project, where each team is developing their own unique application, when a team reaches a point where they do not understand what they are doing and where they are going, they seldom recover.

2. Following on from the previous concern, in many teams, perhaps even the majority, the design “blue print” lacks the understanding and rigor required to code from directly and in many cases the design model is reverse engineered from the working system. Regardless of whether this is a result of ignorance or lack of effort, the path from user requirements to systems implementation has been broken and the integrity of the development process compromised.
3. One of the major problems with projects in 3rd year is the extent of scope. Students are encouraged to limit the size of their applications and to focus on quality rather than quantity. The problem is that teams are also challenged to strive for working solutions (exhibiting key functionality) that include interesting and challenging technology such as the Internet and mobile commerce.
4. Students often do not have the skills and experience to survive the interpersonal dynamics of their groups and therefore do not reach the performing levels identified by Tuckman (in Schwalbe, 2006). Hence the sum of their creative efforts and diversity is not a resultant outcome.
5. The varying skill levels amongst students of a group average out performances which may benefit some, but may negatively impact the higher achievers.
6. Coping with sufficiency of communication and modelling the pragmatists in the teams insist on having.

In the fourth year of the IS major, systems development again constitutes a major portion of the overall curriculum. Following the trend from year three, students form groups and develop significant systems over an 8 month period. The project differs from third year in that the groups are free to select their own application theme, programming environment and implementation platform. This flexibility is encouraged because the department encourages investigation of new application areas and technology. The project deliverables are limited to major milestones to allow groups to function and manage themselves. Minimal scaffolding is provided to the students, although support and expertise is available if student require. This follows the trend from second year of providing groups with less structure allowing them to detach and take ownership of delivery and management of the task. Similar problems are encountered at 3rd year:

1. Like prior years, development teams are self selecting and generally cluster by academic prowess. Without the weekly workshops and deliverables, the disparity across groups intensifies as the weaker groups often lack initiative and work ethic to perform optimally.
2. In many cases the systems are functionally elegant and appear robust. Often however, like the third year projects, the logical design is not translated into a rigorous physical specification and systems are coded directly from requirements. The main weakness in

the overall development process is that students do not development interaction diagrams to the detailed level required and, as a result, their methods are not derived from this process. Tiered architecture, class responsibilities and even the classes themselves are often coded with little reference to the detailed design.

This section has described the planned discourse of the development stream of the IS major. The following sections show how the above curriculum design is a core component of the methodology in an action research process the researchers are engaged with. The Action research process has as core objectives to reduce the trauma of the process and with each year improve the quality of the teaching process to ensure capable graduates are produced.

RESEARCH APPROACH

The researchers are using this research project to provide a reflective process to inform the constant reassessment and redesign of the IS Major. Their knowledge of the area of application and the effectiveness of the methodology and the framework of ideas are being constantly reflected upon as the research process progresses, and other sources of literature are consulted. The reflection and learning is a constant process managed through collaboration amongst the researchers which take the format of regular formal meetings and informal discussions.

This form of research is acknowledged as action research. A key attribute of this form of research is that the researchers are immersed in the research and learning process and are both affecting the research process as well as being affected by the experience acquired. According to Checkland and Howell (1998:23) before embarking on action research, explicit mention has to be made of the: Framework of Ideas, Methodology to be used and the Area of Application.

Consciously knowing these elements of the research approach improves the quality of the action research process and enhances the learning gleaned from the research. The interaction of the elements is shown in Figure 1.

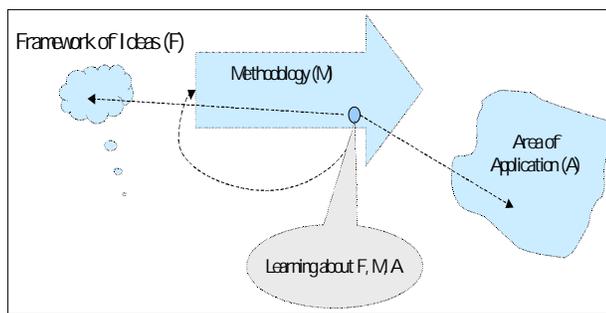


Figure 1 : Elements of the research (Checkland and Holwell, 1998:23)

Framework of Ideas (F)

The following ideas contribute to the framework of ideas influencing the design of the methodology action research methodology:

1. Developers often transcend from practicing traditional development processes to a stage where they detach from the theories they have learnt. Developers are able to do this as they find easier, satisficing approaches to cope with design and communication in a software development environment. They finally transcend from this detaching stage to become fluent practitioners. The fluency stage is characterised by being focused on delivery of working systems instead of process and methods (Cockburn, 2002).
2. Software development can be described as a resource constrained game (effort) where the primary goal is delivery, and the secondary goal is setting up for the next development effort (game) (Cockburn, 2002).
3. Tuckman's group dynamics model of *forming, storming, norming, performing* and *adjourning* is an approximation of the stages of group social dynamics. If this framework is borne in mind much of the trauma of the group dynamics is made more bearable (In Scwalbe, 2006)
4. In the procedural development paradigm, transitioning from the analysis model to the design model is an ever widening chasm. Practitioners have difficulty in explaining how the analysis model is mechanistically transitioned to the design model, perhaps because the rules are internalised (Rosenburg and Scott, 2004). While the UML has provided the OO developer a seamless transition from analysis to design, the "chasm" has moved rather than been eliminated. While the OOAD process provides a clear path between the modelling of object interaction and class responsibility, student seem unable to find the right level and rigor in design sequence diagrams to provide a class design rigorous enough to develop quality object oriented code.
5. Students become disillusioned with the lack of correlation between rigorous documentation and modelling and successful outcome.
6. The multitude of available information, and communities of practice marketed via the internet confuse students. Confusion often tends to come from the convincing arguments which propagate a single approach as a silver bullet.
7. In IS the hype around CASE tools, generalizing practice, and focusing on automation results in less emphasis on communication and invention. Communication and invention however are hugely significant in software development (Cockburn, 2002).

8. Higher education teaching is gradually drifting further away from IS practice (Burns and Klashner, 2005) probably because of a lack of teaching agile methods.
9. IS applications are complex problems and most student groups students often lack the experience and maturity to develop rigorous, “industrial strength” solutions.

Action Research Methodology (M)

The methodology for the action research project is the planning and execution of each of the planned discourses for the major modules making up the IS development stream at UCT. The content, assessment and feedback are carefully constructed from the framework of ideas. Execution comprises the lectures, prescribed readings, the tutorial assessment process, group milestone deliverables, class presentations, and the regular progress meeting with project team mentors. In essence the overall curriculum design of the development stream of the IS major is the core methodology to transcend graduates with newly acquired skills in the direction of rigorous and relevant systems development practices.

Area of Application (A)

The focus of this study is on the 3rd year systems development project and its accompanying project management module. The area of application is expansive and most of the current issues and concerns discussed above present itself during this development process.

LEARNING ABOUT F, M AND A

The action researchers hold regular meetings to reflect on the group development projects to assess how students are coping. A discussion takes place prior to the hand-in of each milestone deliverable. The primary aim is to discuss the design of each: deliverable and the assessment process. After assessment the researchers share experiences and search for opportunities for improvement in the discourse content, methods of teaching and the assessment process. Sharing experiences also highlights students who are transcending very well and can be challenged to improve and those requiring individual support.

Our recent experiences have highlighted the need for the following to be considered:

1. Adding content to establish critical reasoning and problem structuring skills. These skills are crucial during the initiation and planning processes where there is a high dependency on progressive elaboration.
2. Adding a design patterns discourse at third or fourth year level to give students a framework to appreciate the relevance of the knowledge areas in this domain.

3. Moving development projects to an iterative and incremental life cycle. While the concerns voiced previously about the need for mature developers, more staff expertise and contact time are substantial, there are also valid reasons for adoption. Iterative development will enable teams to “de-risk” their projects early in the life cycle. They could build a small number of features in the first iteration and then incrementally build the balance of the system. Small increments allow the team to focus on a small number of use cases which can more easily be designed (and assessed). Problems in terms of specification, logical and technical architecture and operating environment can be picked up very early on in the project. Hopefully by mid-term the project architecture will be stable and the team can focus on delivering functionality.

Given that the fourth year students are the most experienced and mature IS majors, the department have introduced an iterative development approach to their 2006 project. Fourth year teams are required to deliver their system over a minimum of three releases and on three specific dates. In addition they are encouraged to take an agile approach to the modelling and documentation aspects of their project.

4. Supplementing the life cycle with business process modeling. Students grapple to interpret and document their initial introduction to the business problem. The literature shows that both the efforts of Rosenburg and Kendall (2001) on ICONIX and Christian (2001) on UMM are considered methodological enhancements (in Shoval and Kabeli, 2005). Our efforts at UCT stem from similar reasoning. At UCT context and activity diagrams are prescribed during the project initiation phase and package diagrams during the planning phase. This intervention was necessary to assist students improve their overall understanding of the business context, and to also improve the manner in which students articulated the software architecture. A larger view is necessary because students have to introduce parts of working systems whilst still ensuring that integration with the encompassing business context is retained.
5. Resolving the issue surrounding the lack of rigorous design artifacts. This problem can be resolved in a number of ways. A more conventional approach can be put into the teaching of design artifacts, in particular interaction and design class diagrams. This can be supported by reducing the project scope to ensure breadth is replaced by depth. This process should begin in second year where students are introduced to the detailed design process. An alternative approach is to take a more agile perspective. Most agile methodologies would ignore these detailed “implementation” specifications on the grounds that experienced developers understand the design patterns required and can easily work from the

analysis artifacts. This approach opens up another controversy. Perhaps, Lhotkas(2003) direct approach to the design of component-based, scalable, logical architecture (CSLA.NET) is a pragmatic way of transitioning from analysis into design. A comparison of the outcomes of Lhotkas(2003) approach to Bennett et al(2006) is necessary to judge the benefits of either approach.

CONCLUSION

While much is written about the skills required to develop information systems and the evolution of appropriate tools and techniques, there is little to guide academics as to the most appropriate teaching strategy to adopt in the preparation of IS majors. This article focuses on the intervention process used at the University of Cape Town to understand and analyse problems in the teaching of information systems development stream and the impact of interventions.

The IS community is drifting from the assumption that rigorous modelling and application of a standard process is highly correlated to a successful outcome. The experiences at UCT on this action research project and recent literature show that modelling should be more discretionary. At times models are used as conceptual devices to communicate and share understanding and at other time models have very little purpose or impact on the job at hand.

The belief of refining a model until software emerges is proving to be uneconomical and impracticable. These developments are a result of the emergence of agile approaches. Agile approaches emphasise the high dependency on innovation and communication by project participants. People find economic and more efficient ways of achieving outcomes consistently. This pursuit as well as the diversity of IS project teams challenges the existence of a single process such as UP or SSADM.

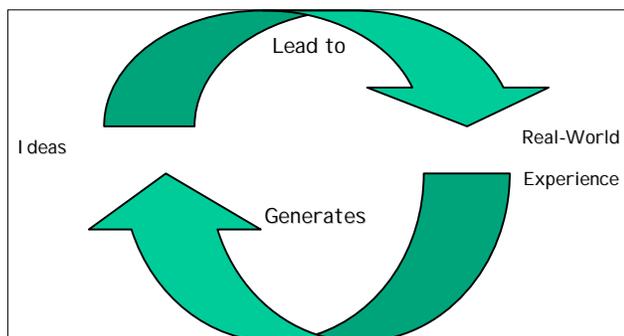


Figure 2: “The Never-ending Dance of Ideas and Experience” (Checkland, 2002)

At UCT we have seen Cockburn’s (2002) model of Learn, Detach and Transcend operational as students question the texts when they come to terms with the factors that influence results. Each year a new batch of students

passes through the system. Figure 2 shows a never ending cycle of Ideas which lead to real world experiences which generate further ideas. The IS major at UCT is under constant scrutiny to make the imparted discourse a lifelong learning experience. The IS world is characterised by hype and temperamental technologies and without a principle centred education IS graduates are unlikely to survive the confusion and contraction, which our discipline abounds with.

For most of the project life cycle the IS artefact is intangible and dependant on human activity. These two fragile dependencies make each IS project a very unique experience. The action research process helps put into perspective the constant flux we academics find ourselves in but is a usefully process to make sense of our experiences as mentors and lecturers in the software development thread of the IS curriculum.

ACKNOWLEDGMENTS

We would like to thank all participants in the software development thread of the IS major at UCT, as well as all the faculty members and students who have contributed their learning experiences so unselfishly.

REFERENCES

1. Bennett, S., McRobb, S., & Farmer, R. (2006). *Object Oriented Systems Analysis and Design using UML* (third ed.). London: McGraw Hill.
2. Burns, T., & Kishner, R. (2005, October 20 -22, 2005). *A Cross-Collegiate Analysis of Software Development Course Content*. Paper presented at the SIGITE'05, Newark, New Jersey, USA.
3. Checkland, P., & Howell, S. (1998). *Information, Systems, and Information Systems*. West Sussex: Wiley and Sons Ltd.
4. Cockburn, A. (2002). *Agile Software Development*. Pearson Education, Inc.
5. Hoving, R. (2003). Executive response: Project Manament Process Maturity as a "secret weapon". *MIS Quarterly Executive*, 2(1), 29-30.
6. Hughes, B., & Cotterell, M. (2006). *Software Project Management 4th Edition*. London: McGrawHill Companies.
7. Lhotka, R. (2003). *Expert One-on-OneVB.Net Business Objects* New York: Apress.
9. Nelson, R. (2005). Project retrospectives: Evaluating Project Success, Failure, and eveything in between. *MIS Quarterly Executive*, 4(3), 361-372.
10. Olson, D. L. (2003). *Introduction to Information Systems Project Managment 2nd Edition*. Singapore: McGrawHill.
11. Richardson, G., & Butler, C. (2006). *Readings in Information Technology Project Management*. Boston: Thompson.

12. Rosenberg, D., & Scott, K. (2004). *Use Case Driven Object Modelling with UML*. New York: Addison Wesley.
13. Schwalbe, K. (2006). *Information Technology Project Management 4th Edition*. Canada: Thompson.
14. Shoval, P., & Kabeli, J. (2005). Data Modelling or Function Modeling - Which comes first? An experimental comaprison. *Communications of the Assoication for Information Systems, 16*(2005), 831-847.
15. Siau, K., & Tan, X. (2005). Evaluation criteria for Information Systems development methodologies. *Communications of the Assoication for Information Systems, 16*(2005), 860-876.