NEW PERSPECTIVES ON TEACHING AND LEARNING SOFTWARE SYSTEMS DEVELOPMENT IN LARGE GROUPS – TELECOLLABORATION

Zenon Chaczko
Senior Lecturer
University of Technology Sydney

David J Davis
Senior Associate
University of Technology Sydney

Craig Scott
Associate Professor
University of Technology Sydney

ABSTRACT
This paper outlines new perspectives on the teaching and learning of software systems development in large groups based on a newly designed, final year, Software Systems Analysis (SSA) and Software Systems Design (SSD) subjects within the undergraduate Software Engineering Program at the Faculty of Engineering, University of Technology Sydney (UTS). Topics described by this paper include shared experiences in implementing these subjects over the last five semesters, discussing their successes, as well as common problems experienced and anticipated. While SSA and SSD subjects are practice based, its formal content is synthesized from other subjects taught within the Software Engineering Program and as practiced in industrial internships, defined within a lifecycle context. The aim of SSA and SSD is to bridge the gap between students’ individualistic, often unplanned, unbudgeted and uncoordinated efforts, that dominate traditional approaches to teaching and learning software subjects; and the demands of planned, process driven, well coordinated and budgeted, modern, large team based development of complex software systems, demanded by industry (vocational expectations) – essentially the Quality of Collaboration (QOC), dependent on the Quality of Project (QOP), Quality of Group Structures (QOG), Quality of System Architectural and Design Frameworks (QSAF), Quality of Instruction (QOI), Quality of Problem Abstraction (QOPA) and Quality of Assessment (QOA). SSA and SSD are, by their nature, evolutionary. But the most critical aspect and challenge is to maintain the careful balance of academic and pedagogical interests with ever changing vocational expectations and practices, utilizing various communication techniques including new technologies such as Tele-Engineering which leverages basic web technologies and enforces a high QOC as well providing a future “distance learning” medium.

KEYWORDS
Software Systems Development in Large Groups, Software Systems Analysis (SSA), Software Systems Design (SSD), Software Development Life-Cycle (SDLC), Tele-Engineering, Quality of Collaboration (QOC), Quality of Project (QOP), Quality of Group (QOG), Quality of System Architectural Framework (QSAF), Quality of Instruction (QOI), Quality of Problem Abstraction (QOPA) and Quality of Assessment (QOA).

1. INTRODUCTION
For a number of years, tertiary software education has been a collection of distinct subjects including software programming, object oriented design, database theory, even project management. Rarely were they
unified into a coherent whole, which defines a measurable profession demanded by industry (vocational expectations). Software Engineering as a “new” profession meant a new approach was needed, so individual software subjects could be used as basic stepping-stones to a more demonstrable and measurable profession. Accordingly, the University of Technology, Sydney (UTS), has introduced a new senior undergraduate pair of subjects called Software Systems Analysis (SSA) and Software Systems Design (SSD) within the Software Engineering degree program. The subjects primarily concern themselves with problem definitions (Jacobson, 1999 and Kleinfeld, 1991), formal and informal requirements analysis; system/software architecture, high and low level design, coding; testing, integration, validation, delivery and maintenance of software based systems, of medium to large-scale complexity, leveraging existing and future web and other software technologies. The subjects demand discipline from the participants, in context with “process”, but permit flexibility in adapting modern “modeling” methods to software system project development. Other educational interests required some participants to have a theoretical knowledge of software development processes/methods/paradigms without actually physically implementing a software system.

The real driver for these subjects was a vocational expectation, industry-based demand for software graduates to have “real experience” within large groups, being project focused, process and outcome driven, and permitting experimentation with hierarchies and other group structures – essentially providing “peer collaborative” experience. This demand had to be balanced with academic, pedagogic, technical rigor, depth and formalism interests. To achieve this, both subjects were designed to complement each other. First by unifying theory (SSA) into a Software Development Life-Cycle (SDLC) with outcomes focused on the depth of research, scholarship, problem definition/solving (Kleinfeld, 1991), analytical skills, critical and reflective thinking coupled with social and communication skills including new group formations, group participation, experimentation with hierarchies, role playing and role rotation. Most critical of all was the development of the individual as a confident, coherent, ethical, engineering professional. Whereas SSD fulfills vocational expectations by applying theory and personal development, as engendered by SSA, into an implementation base focusing principally on delivery, development processes, schedules and budgets which results in quality, maintainable, reliable and usable software system products. Hence, SSA was designed to be the Software Engineering “theory”, allowing an exit point for those participants who required theory only and SSD the Software Engineering “practice”.

This paper investigates and reports on our experiences of teaching complex software system development, within large groups, across the SSA/SSD pair. It aims to evaluate the outcomes of this experience with the object of following a reflective process (Goethals et al, 1997) at each subject conclusion to improve learning and teaching. It will describe methods applied in the teaching and learning framework model, which combines best practice aspects, engineering standards, pedagogy, project management, economics (costs/finances) (Boehm, 1988), the problem domain, paradigms, stakeholders concerns, technology and tools. The model can be best described as a multi-dimensional (Jacobson, 1999), reflective approach to teaching and learning (Odeh, 2003). By using this approach, reflections on the effectiveness, efficiency and comprehension of issues involved can be efficiently analyzed and resolved.

Next, the paper provides observations on the outcomes of a complex software system project development in SSA/SSD. In particular the Tele-Engineering project, coupled with a reflection on the quality of collaboration seen as vital to the success of any project. The observations were derived from quizzes, content of project packages, student comments on tools and technologies used, group presentations of implemented systems, exit interviews, product demonstrations, group-project coursework, discussions on student evaluation, feedback from project stakeholders: students (in various project roles), tutors, lecturers, subject matter experts, customers and invited guests from industry.

Finally, a range of future research and development topics is discussed followed by a conclusion, including a summary of the outcomes of this report.

2. TEACHING AND LEARNING METHODS IN SSA AND SSD

Group-oriented, practice based SSA/SSD projects are carefully constructed to guarantee all participants balanced, equitable, structured opportunities and tasks in which to apply project management, processes, methodologies and techniques introduced by lectures, presentations, tutorials and other training sessions. The
most critical aspect of the teaching and learning tasks is to maintain a careful balance of academic and pedagogical interests with ever varying vocational expectations and industry practices.

Subject participants are required to make a single choice from a provided short list of telecommunication, intelligent control and resource allocation management related software projects which originate from industry and/or academic research. The projects involve disparate technologies, often in distributed network environments. There are, of course, a variety of engineering standards, processes, methodologies, technologies and design options the participants have to make, when selecting their choice. Each of these selections plays a vital role in the project’s success.

Each class is divided into large project groups, up to 20 participants, which are further sub-divided into a number of smaller teams (usually 5 to 7 participants), dependant on group selected management and resource allocation plan strategies. Each team usually takes responsibility for one of several major deliverables, but one team asserts project coordination (management) and systems engineering roles.

SSA introduces processes, paradigms, tools, technologies, design concepts coupled with group participation and individual development. The role of the instructor in SSA is that of teacher, customer, project facilitator and mentor.

SSD adopts a more laissez-faire and "constructivist" approach (Wilensky, 1995 and Brown et al, 1989), where knowledge emerges as a result of collaboration. Within the SSA/SSD project context, learning is seen as a developmental process first occurring between individuals (socio-cognitive domain) and later, within the individual (intra-personal domain). It is recognized that teaching and learning in SSA/SSD shall remain a constructive activity requiring dynamic reorganization of existing concepts, methods, approaches and projects needs. This is reflected in the fact that participants are expected to form and manage their own groups, adopt their own methods, processes and paradigms (based on training received in SSA) within the SDLC. In fact, they take total control and responsibility for their own learning. The only compulsory aspect is that the project groups are required to deliver a quality product, on schedule, within budget and with due process as determined by the project size and complexity. The SSD instructors are principally customers, with subsidiary roles such as mentors, assisting the optimization of individual efforts.

Figure 1. Adapted Activity Focus Method (AFM) vs. Spiral Model (Avision et al, 2000) for Teaching and Learning.

The AFM method, Figure 1, combines best software engineering practices and the Spiral Process Model (Boehm, 1988) with recommended series of steps, procedures (Avision et al, 2000) and activities applied at changing intensity levels within SSA/SSD which occupy two semesters (one semester per subject).

SSA/SSD use a web-based learning Telecollaboration tool: UTSOnline, for: debating (discussion boards), email interaction, keeping development material up to date, accessing learning resources and submitting assessment tasks. UTSOnline is used as a repository of all project artifacts and permits the continuation of project focused activities. The success of UTSOnline for group communication and collaboration leveraged the explorative extension of web based tools and technologies into a fully fledged Tele-Engineering environment, providing subject participants with virtual classrooms, laboratories and advanced web-based communicative tools within an integrated, collaborative environment. It was perceived that this extension would directly assist distance learning without disadvantaging remote participants. The introduction of this technology would have other major advantages: first, by assisting participants in the act of collaboration, by
imposing requisite rigor and discipline and second by compelling participants to adopt and streamline collaborative processes, without compromising the modeling domains and pedagogical concepts, currently employed by the SSA and SSD subject pair.

Tele-Engineering, as a new technology, has many promises and pitfalls which will eventually be overcome as the constituent technologies are advanced and mature. As an introduction, the technology was set as a software system project in the Spring Semester, 2003, for SSA mainly by leveraging existing UTSOnline Telecollaboration facilities, adding multiple interactive technologies including video conferencing, “chat room”, “white-board” and shared application databases. SSA resulted in a Software Requirements Specification, Software Architecture and High Level Design which provided a preliminary review of the technology. The Autumn Semester, 2004, saw the project further developed to include a fully developed proof of concept – including revised software architecture/designs and an implemented system on intended platforms, interconnected within a minimal configuration (maximum of three collaborators).

3. MODELLING DOMAINS AND PEDAGOGICAL CONCEPTS

SSA/SSD share the teaching and learning domain model as depicted at Figure 2. The model highlights the multidimensionality of large group-based complex software system development activities by underpinning domains such as: engineering best practice aspects and standards, stakeholders concerns, pedagogy, design methodologies, economics, technology and tools as its base (i.e. the problem domain). The depiction of the problem domain as the model’s base, aims to reflect the fact that a well-designed and implemented software system has to be well understood. The communication domain plays a pivotal role and underpins other domains of Software Engineering including: project management, collaboration (ethics, work culture, sociology and psychology), processes, technical and organizational abilities, and skills; which underly a successful project delivery. The model’s best practice domain highlights quality management aspects, involving quality of process and quality of deliverables. To ensure quality of the final product, participants are obliged to identify and analyze the project’s Key Performance Areas (KPA) including Risk Management, Configuration Control, Traceability Models, Fault Tracking mechanisms and Project Reviews. Participants are obliged to discuss, manage and monitor the project budget, always keeping in mind affordable solutions (optimal cost). Needs and views of all stakeholders are considered, tracked and managed.

The pedagogical dimension (as depicted in the model for learning and teaching, at Figure 3), measured in outcomes (Adelsberger et al, 2000), includes mandatory aspects of the project’s background study, problem scope and problem understanding, definition and analysis, critical thinking involving reflective exercises and self-assessment, stakeholder identifications, and interactive decision making - involving all SSA/SSD project stakeholders. This is supported by participants making choices (i.e. tailoring) of effective engineering standards, processes, existing and future technologies and tools; discussing and presenting options to be considered for an architectural/system design and finally selecting the best solution according to pre-defined and reviewed criteria. The pedagogical aim is not only about discussing and "covering" a variety of software system development issues but to include the integration of software systems knowledge, methodologies, economics, throughout the SSA/SSD curricula.

SSA/SSD adheres to the Pedagogical Concepts Model, Figure 4, which is a modified version of the Essen Learning Model (ELM) (Adelsberger et al, 2000). In this model, the participants “act” in various developed scenarios. Acting in multiple, changing roles permits participants, including instructors and other teaching entities, conceptual growth and permit a deeper understanding of the topics, aimed to last throughout their professional careers. Such modeling is designed to encompass context (experience, knowledge, environment, application) and content (didactic units, composite didactic units, sequence, steps, implementations).
Figure 2. Domain model of teaching and learning in SSA/SSD

Figure 3. Model for learning & teaching SSA/SSD (modified model of IEEE Learning Technology Standards Committee)
4. CONCEPTS OF THE TELE-ENGINEERING SSA/SSD PROJECT

The Tele-Engineering SSA/SSD project (TEPS) comprises two main aspects: Virtual Meeting Places (Venues) and Virtual Interaction (Features) including: Sight (Video Conferencing), Sound (Audio Conferencing), shared Drawing Space (Doodlepad), Textual Space (Chat) and Agenda/Date Book (Scheduler). The project implementation focussed on the ConferenceXP technology which incorporates a suite of low level protocols and tools (APIs) to assist in network communication, created for the Microsoft .NET Environment and which was largely untried and untested.

Operationally, a collaborator could select and join a venue (Virtual Locale) from a list of venues (maintained by a Server - centralised or distributed). Once “at” a venue, a collaborator could then activate a session with one or more collaborators utilising one or more of the Features (video, audio, textual space, drawing space, agenda/datebook). The Tele-Engineering Server (TES) has minimal functionality, including the maintenance of venues, permitted users and optionally shared databases. Another aspect is an administration feature which handles the attachment/detachment and priority levels of the collaborators which is either centralised by TES or distributed on one or more collaboration clients. The conceptual structure is shown at Figure 5 and a product perspective at Figure 6.
The TEPS architecture uses a fat-client/thin server design which places its emphasis on client side services. The design is user-interface focussed, due to the fact it uses existing technologies for most of its applications. Boundary modelling identified: TEPS Core and UI, Chat, Audio and Video, DoodlePad and Scheduler as mandatory subsystems. The TEPS architecture functions as a layered “framework” (Figure 7) for interfacing between a collaborator’s permission levels and the Features. It includes the capacity to interface with its counterpart on another client without intervention from the TES. It permits insertion, replacement or removal of technologies as the need arises – as technologies are developed or improved.

Figure 6. TEPS- a product perspective.

Figure 7. TEPS - Subsystems using Conference XP.
Project difficulties included the experimental nature of the ConferenceXP technology which was found to be both preliminary and poorly documented; APIs had to be prototyped before they could be implemented – mainly to discern their nature and any potential defects; and even Pentium 4, PC platforms had insufficient CPU capacity to concurrently activate more than two Features.

5. EVALUATION OF LEARNING AND TEACHING OUTCOMES

Academic and pedagogical achievements within the SSA/SSD projects need to be measured by more rigorous analyses (Jasper, 1997) including interviews, questionnaires, opinion scores and surveys (the Subject Feedback Surveys are used at UTS). The more substantive analyses would assist the evaluation of the students’ ability to associate ideas and to draw conclusions. However, we were keen to compare the SSA/SSD results with those of CSA/CSD (Leaney et al, 1996). During exit interviews we performed the evaluation of students’ scores using the SOLO taxonomy of Biggs (Ramsden, 1992). In most cases we could confirm there was a strong correlation between high SOLO scores and individual participants who score highly overall in the project. Quality of Collaboration (QOC) was also a factor for individual participants who score highly, and included the success of project delivery and the success of Tele-Engineering as a concept for distance instruction.

A QOC framework was derived from the observation, that: “When work is truly collaborative…partners make contributions to the decisions that are made, partners take note of and seek out each other’s opinions and suggestions”, Abnett et al (2001). In addition, the SSA/SSD assessment procedures (including exit interviews) showed that QOC is dependent on the Quality of the Project (QOP), Quality of the Group (QOG), Quality of the System Architectural Framework (QSAF), Quality of Instruction (QOI), Quality of Problem Abstraction (QOPA) and Quality of Assessment (QOA).

It was found that a project must be “attractive” to individual participants – the challenge of using “new” technologies and Tele-Engineering as a concept for distance collaboration ensured participants complicity in the collaborative process. Thus the ingredients of discernible, rigorous engineering content, working with “new technologies” (vocational expectations) and a certain amount of social relevance is the correct mix for a high QOP. Setting up project groups by random selection of participants from the subject population, produced a higher QOG than by participants selecting the project and whom they work with. Working with “strangers” versus working with friends tended to raise the QOG – “friends” tend to, as Abnett et al (2001) noted, “…dominate … other(s) with more instructions and directions and fewer requests for opinion.”. QOG also depended on an adopted leadership model. Student group leadership tends to two extremes: the inflexible hierarchical or the completely democratic. The former, also known as the dictator model has a sense of the “false prophet syndrome” where a few participants seize power and coerce others to follow their direction, often with dire results such as poor or no deliveries. The latter extreme invariably led to chaos and poor quality deliveries. In practice, higher QOG was achieved somewhere between the extremes and as modified by the variability of interests within the group itself (what is practical).

Various methods of instruction were tried including conventional lecture/tutorials, self-instruction and mentoring (laissez faire model). Higher QOI was achieved in SSA by conventional lecture/tutorials, supplemented by self instruction such as small research projects which were presented by individual participants – on the principle of value adding to the theory taught. In SSD higher QOI was achieved by mentoring, rather than formal instruction, and self-instruction where one or more participants research a technology and fed-back their findings to the group by mentoring other participants – for the Tele-Engineering Project implementation, this was particularly useful, particularly for technologies such as ConferenceXP and .NET. SDLC provides “stepping stone” processes in which one project step leads inextricably to another. However, SSA/SSD projects often have these steps in parallel. This meant a flexible Systems Architectures (high QSAF) coupled with problem abstractions using boundary modeling (QOPA) was vital for a high QOC. Finally, QOA as measured by the participants’ perception of fairness and equity, meant a high esprit de corps which led to a better QOC – assessments included “deliverables”; personal logbooks, reflections and presentations; project artifacts that indicated adherence to process; exit interviews; mentoring interviews; instructor observation and occasional quizzes.
6. FUTURE RESEARCH AREAS

In the course of our research, we observed, in general, there is a shortage of useful general-purpose models (which include best practices, industrial standards, human communication models, technologies, stakeholders, economics, engineering and scientific methods and evaluation metrics) available for problem-based (Warhurst, 2001) teaching and the learning of complex software systems engineering in the academic environment. Particularly, measuring and modeling QOC, QOP, QOG, QSAF, QOI, QOPA and QOA. There are also many questions related to learning and teaching of complex software systems development, including:

1. Are the teaching and learning models in SSA/SSD optimal, if not - how can they be evolved?
2. How can constructivist models (Wilenksy, 1995) be better adopted for learning and teaching in large groups developing complex software systems and how might these models be implemented within the environment of tertiary institutions?
3. What changes in educational programs and processes would need to be adopted for instructors to deliver quality subjects that prepare students for an SSA/SSD experience?
4. Are there specific issues and patterns that relate to the appropriateness of particular methods in teaching and learning QOC, QOP, QOG, QSAF, QOI, QOPA and QOA concepts at various technical and educational levels?
5. How might the university education faculty assist and benefit from our innovative pedagogical approaches to teaching software systems development within large groups?
6. How can our perceived success lead to the development of new pedagogical tools suitable for software engineering programs that include software systems analysis and design curriculum.

Several researchers have focused on practice based methods for teaching software engineering in large groups (Odeh, 2003), methods of teaching complex systems concepts (Jacobson, 1999) as well as methods of teaching collaborative engineering (Favela et al, 1997), but very few seem to have undertaken studies on how to effectively teach and make sense of the relationship that exists between large group based strategies such as QOC, a multitude of participant perspectives and collaborative outcomes within the construction of complex software systems.

7. CONCLUSION

A practice based approach to the teaching and learning of software systems development is fundamental to the success of software engineering educational programs. We demonstrated that a strict approach to following software development processes and managing project budgets has no detrimental effect on the final results of the project. On the contrary, groups, that tried to limit the role of process, developed human communication problems, created negative team dynamics, exhibited disinterest and often developed a “culture of blame” (Brown et al, 1997) thus lowering QOC which directly impacted the projects’ final results.

One of the challenges that we face in SSA and SSD is for our graduates to be able to reverse industry trends towards large software project failures. An indication of the success of the SSA/SSD pair has already been somewhat measured by industry. All software engineering graduates that have completed SSA/SSD have received full time employment within six months of completing their degrees. Results of a more rigorous analysis of students’ progress, as discussed above, were also initially positive.

Cross-disciplinary aspects (Figure 2) and newer ways of teaching complex software system development in large groups provide a challenge and an opportunity to present new perspectives add important dimensions to the increasing amount of engineering skills and knowledge. These are required to be integrated in a consistent manner that is manageable, pedagogic, cognitive, effective, efficient, satisfying and useful. There are many students for whom the sheer challenge of understanding new ideas and solving problems will motivate them to learn more about software systems development. For most students however, we must find new engaging and practical approaches so that they not only understand what they are being taught, but can also acquire generic skills that can be used not only in their vocational experience but also in their daily lives.
In this report, we have presented methods and models that we adopted in SSA/SSD for solving teaching and learning issues. The future focus of software development teaching and learning in large teams will be placed on SSA/SSD acknowledged problems with technical resources, collaborative engineering tools such as Tele-Engineering and support issues. As a closing comment, there is need for a pedagogic vision of a renewed importance for innovative approaches to teaching software development of complex systems by large groups. The new approaches would serve as a means to develop and cultivate what might be a new and principles driven type of software engineering literacy in order to adequately address any emerging issues and problems in the software industry of the 21st century.

ACKNOWLEDGEMENT

Professor Robin Braun, Deputy Director IACT, Venkatesh Mahadevan Senior Research Associate and Shmuel Markovits Senior Research Associate, Faculty of Engineering, University of Technology Sydney, for their expert guidance, support and sometime involvement in the SSA/SSD subject pair.

REFERENCES


Boehm B., 1988, A spiral model of software development and enhancement, IEEE Computers, vol. 21, no. 5, pp. 61-72


Jacobson M. J., 1999, Complex adaptive systems, problem solving, and cognition: Preliminary research towards the design of advanced learning technologies, Final report to the National Science Foundation Applications of Advanced Technologies Program, Athens, The University of Georgia, Learning and Performance Support Laboratory.


