Using SoDIS™ as a Risk Analysis Process: A Teaching Perspective

Don Gotterbarn and Tony Clear
School of Computer and Information Sciences
Auckland University of Technology
Private Bag 92006, Auckland 1020 New Zealand
Donald.Gotterbarn@aut.ac.nz Tony.Clear@aut.ac.nz

Abstract

There are several difficulties we face when showing our students key processes and techniques for software development. In this paper, issues related to teaching students how to manage risks in software projects are profiled. The concepts and process for implementing Software Development Impact Statements (SoDIS) are outlined; with its supporting CASE tool the "SoDIS Project Auditor" being described. Different ways of applying the SoDIS process and the CASE tool are demonstrated, through some brief illustrative case studies. The paper suggests ways of using the process and the tool to enhance teaching in computing courses including software development projects, software engineering, project management, ethics and professionalism. This work occurs under the umbrella of the SoDIS SEPIA collaborative research programme which aims to promulgate use of the SoDIS process, in both industrial and educational computing spheres.

1 Introduction

There are several difficulties we face when showing our students the key processes and techniques for software development. Traditionally Computer Science courses are high content/ low context. You learn how to apply a tool without understanding how its context should effect the way in which it is developed and designed. This instrumental perspective also leaves students ill prepared for considering both the impacts of their work, and the parties who may be affected by what they do. Risk analysis is a commonly taught technique to broaden assessment of projects and consider multiple dimensions. However traditional quantitative risk assessment models fail to take into account impacts on stakeholders, and numeric calculations may often trade-off very legitimate human concerns.

A common, but disconcerting aspect of software development classes is that the appeal to the standard risk analysis techniques is not generally accepted by students. For many students, especially at undergraduate level, the challenge of applying systematic and sustained risk management processes on top of their often poorly conceived and maintained project plans is simply too big a step to take. This reluctance to apply risk analysis is compounded by what they hear about software disasters for systems that employed these risk methods. Further, even in their own experience, these risk methodologies did not mitigate problematic results. As a result students fail to understand the relationship of risk to each stage of a project and its manifestation in another if left unaddressed. They also need to understand that as the problems you are dealing with quantitatively increase they also change qualitatively in complexity raising issues that have not been addressed in the technological side. A more general concern in teaching computing is the need to inculcate a sense of professional responsibility, for instance, when Hal kills who is to blame?

A process called Software Development Impact Statements (SoDIS) helps identify the missed risk items and provides clear evidence of how risks and disciplined approaches at early stages of development has a direct impact on the software quality and scale of problems in later stages of development. The use of the SoDIS process has helped to reduce many of these problems. For example, it use on government plan for e-voting identified 121 previously unidentified risks. It will be helpful to look at the type of problems the process addresses before giving a detailed description of the process and its associated software.

2 Many problems with common causes

Software developers may include new risk processes and reviews into their software development projects but they still encounter project failures. The areas of risks considered typically include what might cause missed schedule, budget overrun (insufficient student resources), and failure to meet the system's specified requirements. These risks are identified using a quantitative analysis. Nevertheless problems continue. It has been shown that a quantitative analysis is inadequate or incomplete as a methodology. Software may be produced on schedule, within budget, and meet all the owner's specified software requirements, but nevertheless fail due to other adverse impacts. Why? Simply because the software developers

chose to focus on a narrow set of quantifiable risks related only to those directly involved in the development of the software and overlooked identifiable qualitative risks.

For example, the Aegis radar system was a success in terms of budget, schedule, and requirements satisfaction; even so, the user interface to the system was a primary factor in the USN Vincennes shooting down a commercial airliner killing 263 innocent people. The narrow focus on certain stakeholders to the exclusion of others, led to developing an interface that was inadequate in a combat situation. Fortunately not all software failures have impacts of this magnitude; nevertheless problems attributable to limited risk analysis are pervasive. Students need to have an appreciation of both sides of risk analysis so they are less likely to be party to such massive disasters.

There are two common factors that contribute to these failures; both related to those who have something to gain or lose as a result of the software project — system/project stakeholders. First, there is significant evidence that many of these failures are caused by limiting the consideration of system stakeholders to just the software developer and the customer (the instructor and the student). This limited scope of consideration leads to developing systems that have surprising negative effects because the needs of relevant system stakeholders were not considered. In the case of the Aegis radar system the warning messages were not clear to the users of the system operating in a hostile environment. Second, these types of failures also arise when developers limit the scope of software risk analysis just to the technical and cost issues. A complete software development process requires 1) the identification of all relevant stakeholders and 2) enlarging risk analysis to include social, political, professional and ethical issues.

A risk analysis method called a "Software Development Impact Statement" (SoDIS) helps to reduce the number of software failures [Gotterbarn, 2001a, b]. The SoDIS process expands existing software development risk analysis methods by adding a qualitative element. It helps developers explicitly identify the relevant stakeholders and broaden the scope of risks anticipated.

2.1 Stakeholder Identification

2.1.1 Projects

Disparate types of software projects have different types of stakeholders. For a project to succeed there must be effective risk resolution that considers the project type, and the opinions of all stakeholders. Different expectations regarding how to judge a project as a success or a failure must also be accounted for..

For example, in a recent case, reported by Davey (2003), in New Zealand, a developer was asked: 1) to develop an Internet filter; which 2) would only allow browser access to web sites which were added to an approved web site list; 3) The filter was to be installed in a school; that 4) was going to network all of its computers. If we merely consider the functionality described in 1) and 2) then one particular set of risks in development would be addressed. The contextualization provided by item 3) changes the risks that need to be considered and the way in which the software should be developed.

Effective development strategies require a full appreciation of the nature of the project. Current thinking using the size and complexity of the project appears limited. There are those who simply categorize projects by size in order to adopt an appropriate approach. These approaches do not shed much light on the risks in the Internet filter project. The problem with these approaches is that they are inward looking, focusing simply on the obvious within the narrowly confined boundaries of the project. This leaves projects vulnerable to unforeseen problems and risks and lead to eventual total failure. Such situations are easy to imagine in the Internet filter example. Risk analysis should be outward looking and take into account the overall environment within which the software will be used and the target application area. . In the Internet filter project, adding the context of a school in step 3 changes the types of risk that have to be addressed in this software development. The type and context of the project helps determine a subset of the stakeholders relevant to a project.

2.1.2 Consequences of identifying particular stakeholders

Misidentified or unidentified stakeholders are a major contributory factor to the ineffectiveness of current risk analysis methods. Recent research has confirmed that inadequate identification of project stakeholders and how they are affected by a project is a significant contributor to the project's failure. Establishing the right project scope is essential in defining project goals. The stakeholders determine the scope of consideration. Normally, the stated needs of the customer are the primary items of concern in defining the project objectives. Investigating 16 organizational IS-related projects led Farbey et al, [1993] to conclude that "... the perception of what needed to be considered was disappointingly narrow, whether it concerned the possible scope and level of use of the system, [or] the range of people who could or should have been involved ..." They discovered, with the exception of vendors, all stakeholders involved in evaluation were internal to the organizations. The reason for this restricted involvement is that these are the only stakeholders originally identified in the traditional project goals or system requirements. This is similar to the way many school projects are developed. For a variety of reasons schools often reinforce this dangerous approach.

2.2 Broaden Concept of Risk to include Qualitative Issues

The concept of 'software failure' and its correlate 'risk' need to be broadened in our curriculum to include qualitative issues. "Software failure" is not simply an issue of schedule, budget and reliability. Software has been developed which, although meeting stated

requirements, has significant negative impacts on the circumstances, experiences, behaviour, livelihood, or daily routine of others. The Internet filter system will promote censorship by omission. The system will require monitoring by the school's Internet censor. The system will limit student preparation for later courses. These qualitative issues with software have been recognized but treated inadequately by both Information Systems and software engineers.

The expansion of the scope of a project to include all relevant stakeholders will broaden the types of risks and issues considered. When considering software development we need to consider the impact of the system as a whole. In the past, the developers have restricted their involvement in the development of a product to its technical elements. This self-imposed limitation has contributed to the development of software that has been inferior and has had negative consequences for others. Students need to be taught that the systems they develop perform tasks that affect other people in significant ways. The production of quality software that meets the needs of others requires both the carefully planned application of technical skills and a detailed understanding of the social, professional, and ethical aspects of the product and its impact on others. Types of risks and issues identified in professional codes of practice, conduct, and ethics can be used to relate particular stakeholders to development tasks.

The SoDIS process applies the distinction between quantitative and qualitative research to risk analysis in a novel way. The way software development tasks are related to stakeholders facilitates the qualitative side of risk analysis. A quantitative approach to risk relies on developing metrics that can be used to describe the risks in terms of dollars lost, days over schedule, number of functionalities not met. Whereas a qualitative approach to risk relies upon a textual description of the risks which can be categorised but not quantified. Risks like over constraining Internet access, security of the approved list are not quantifiable but can be categorised into critical, significant and minor risks. This categorisation provides an understanding of unique types of risks thus facilitating the discovery of appropriate solutions for the risk. The quantitative and qualitative approaches are complementary and both are necessary.

3 The SoDIS Process

The Software Development Impact Statement (SoDIS), a modification of an environmental impact statement, is a way of addressing the need to modify project tasks in a formal way. A SoDIS, like an environmental impact statement is used to identify potential negative impacts of a proposed project and specify actions that will mediate those impacts. The SoDIS process completes risk analysis by addressing a project's qualitative issues. A SoDIS is intended to reflect both software development process and the more general obligations to various stakeholders.

Any phase or aspect of software development consist of a set of things that need to be done to complete that phase, e.g., functional requirements, resource allocation, module testing, code development etc. For the moment we will use 'tasks' as a generic term to describe these.

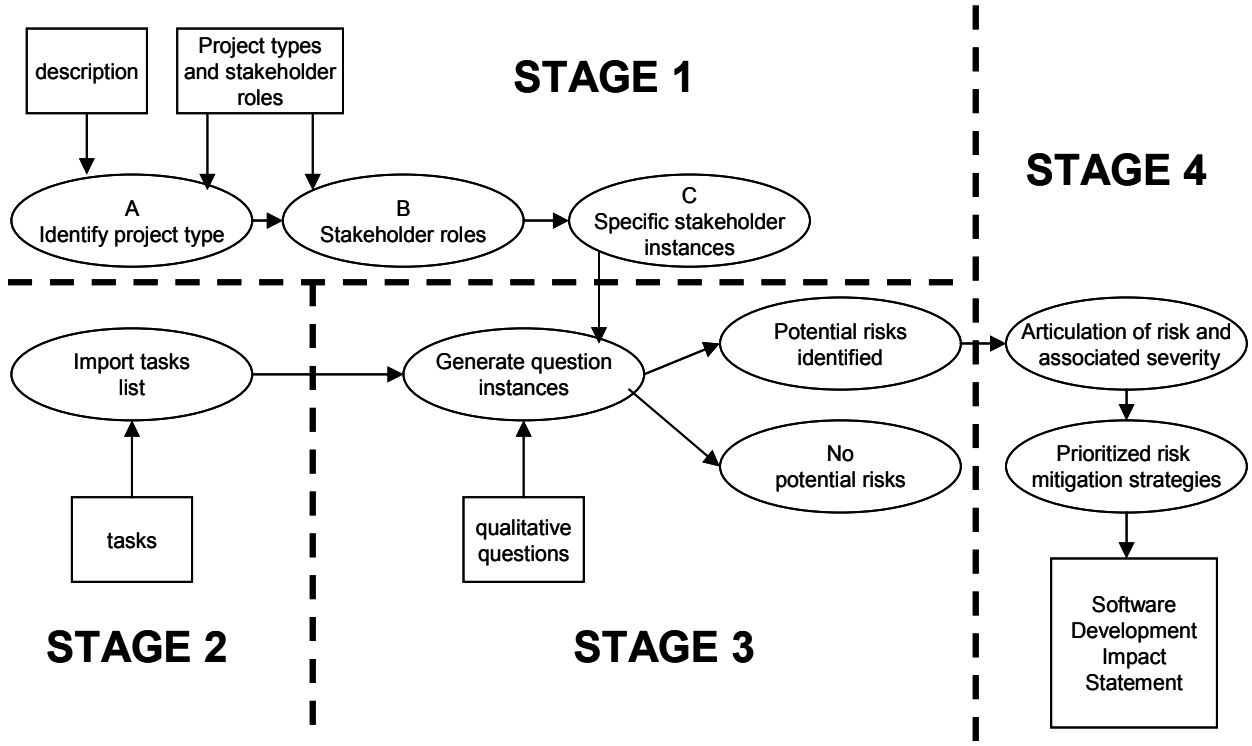

Figure 1: SoDIS™ Process (© 2003 Gotterbarn and Rogerson)

The goal of the SoDIS process is to identify significant ways in which the completion of individual tasks, that collectively constitute the project, may negatively affect stakeholders. It identifies additional project tasks that may be needed to prevent any anticipated problems and identifies changes in some tasks needed to prevent any anticipated problems. Moreover, the intent is to identify these risks in a pre-audit of each software development phase by examining their respective task lists before that software development phase is started.

As shown in Figure 1, the SoDIS process consists of four stages: (1) the identification of the project type together with immediate and extended stakeholders in a project, (2) the identification of the tasks in a particular phase of a software development project, (3) the association of every task with every stakeholder using structured questions to determine the possibility of specific project risks generated by that particular association, and (4) completing the analysis by articulating the concern generated by the associations, determining the severity of the risk to the project and the stakeholder, and recording a possible risk mitigation or risk avoidance strategy. The resulting document is a Software Development Impact Statement (SoDIS) which identifies all potential qualitative risks for all tasks and project stakeholders. Of course figure 1 is simplified for the sake of readability, since the SoDIS process allows for ongoing review throughout the project. Updates to the analysis may be entered as they come to the attention of the reviewer/project manager.

A prototype tool – The SoDIS Project Auditor- has been developed to facilitate this process. The SoDIS Project Auditor keeps track of all decisions made about the impact of project tasks on the relevant project stakeholders and it enables the problems identified to be addressed proactively.

3.1 Task List (stage 1a)

A SoDIS is developed from a task list. Depending on the stage of software development, the task list can consist of a set of requirements, a software design plan, a code development plan, a test plan, etc. The process of developing a SoDIS encourages the developer to think of people, groups or organizations related to the project (stakeholders in the project) and how they are related to each of the individual tasks that collectively constitute the project.

3.2 Stakeholder Role Identification (stage 1 b)

The system provides a standard list of stakeholder roles related to most projects. Stakeholder roles are added to the standard list of roles with each change of project type. For example, a business project will include corporate stockholders, while a military project will not have stockholders as a standard stakeholder role. The system also enables the SoDIS analyst to add new stakeholder's roles and project types.

3.3 Identification of Stakeholders (stage 1 c)

A preliminary identification of software project stakeholders is accomplished by examining the system plan and goals to see who is affected and how they may be affected. When determining stakeholders, an analyst should ask: Whose behaviour, daily routine, work process will be affected by the development and delivery of this project; Whose circumstances, job, livelihood, community will be affected by the development and delivery of this project, and Whose experiences will be affected by the development and delivery of this product. All those pointed to by these questions are stakeholders in the project. The identification of stakeholders must strike a balance between a list of stakeholders that includes people or communities that are remote from the project, and a list of stakeholders that only includes a small portion of the relevant stakeholders.

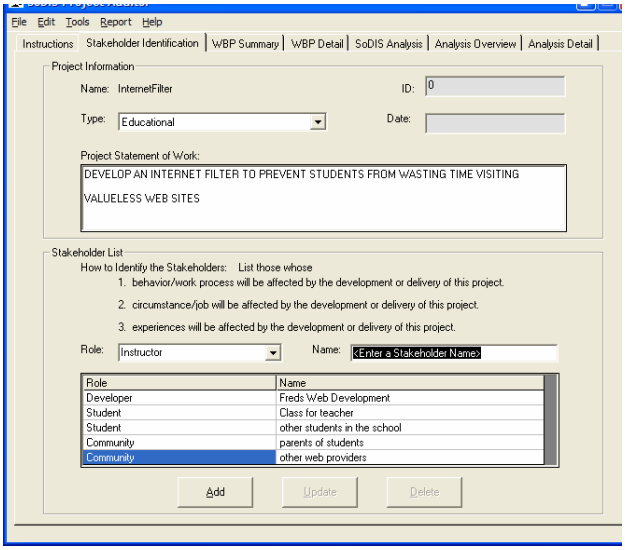

Figure 2: Stakeholder Identification

The stakeholder identification form (Figure 2) contains a Statement of Work that helps remind the analyst of the project goals at this phase and facilitates the identification of relevant stakeholders. The relation between identifying stakeholders and doing a SoDIS analysis is not linear. In the prototype tool the stakeholder form and the SoDIS analysis form are dynamic and enable the iterative process. If while doing an ethical analysis, one thinks of an additional stakeholder he/she can shift to the stakeholder identification form, add the stakeholder, and then return to the SoDIS analysis that will now include the new stakeholder.

3.4 Identification of Tasks (stage 2)

In a project management model the component tasks - "work breakdown packages" - only address the technical issues. These individual task descriptions are used in the reviewing and monitoring of the project. All of these tasks are ordered in a hierarchy of dependency on one another.

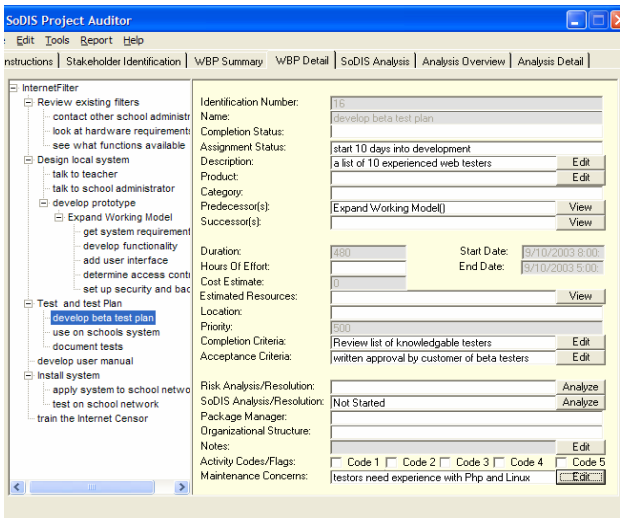

Figure 3: Detail Task Screen

Each of these individual tasks may have significant ethical impact. The SoDIS is used to help the developer responsibly address the ethically loaded potential of each identified task. This is accomplished by including a SoDIS analysis in the standard descriptive elements of the highlighted task (Figure 3). The process is the same for any cluster of task types. A task is highlighted- develop beta test plan- and then details related to it can be recorded or a SoDIS analysis can be done.

The SoDIS analysis process also facilitates the identification of new tasks or modifications to existing tasks that can be used as a means to mediate or avoid identified concerns. The identified tasks need to be incorporated into the task list

3.5 Identify Potential Ethical Issues (stage 3)

The risk analysis ties stakeholders to tasks by raising issues derived from computing codes of practice and conduct. These have been framed as a set of 32 issues which tie a task to a stakeholder in the form of a structured question. The prototype forms the triple consisting of a task and issue and a stakeholder producing a question. The question is placed in the bottom frame of the SoDIS Analysis screen (Figure 4). In this case the developer is asked if the development of a filter for one teacher's class will also limit access by other students.

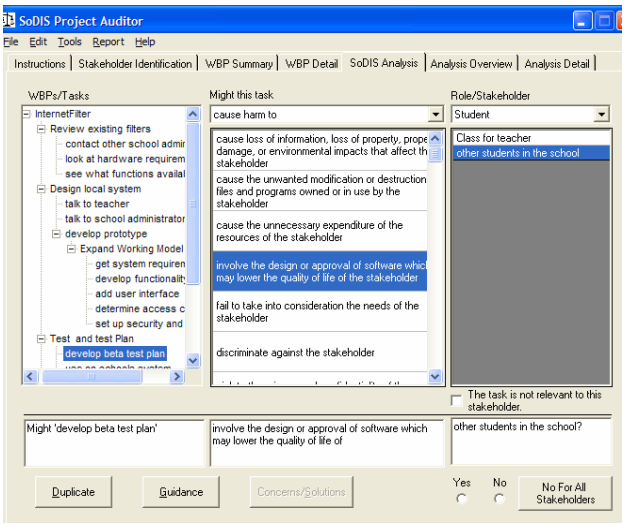

Figure 4: SoDIS Analysis screen

There may be some special circumstances that are not covered by these 32 questions so the system enables the SoDIS analyst to add questions to the analysis list.

3.6 Identification of concern and Mitigation process (stage 4)

When an ethical concern has been identified, the analyst gets an ethical concern form (Figure 5) that asks the analyst to record their concern with the task and record a potential solution. The most critical part of this process is on this form, where the analyst is asked to assess the significance of their concern with the work breakdown package being analysed. This is a judgment of QUALITATIVE impact. If they don't have a proposed solution for their concern at the moment then they can save the record of the concern and go on with the analysis.

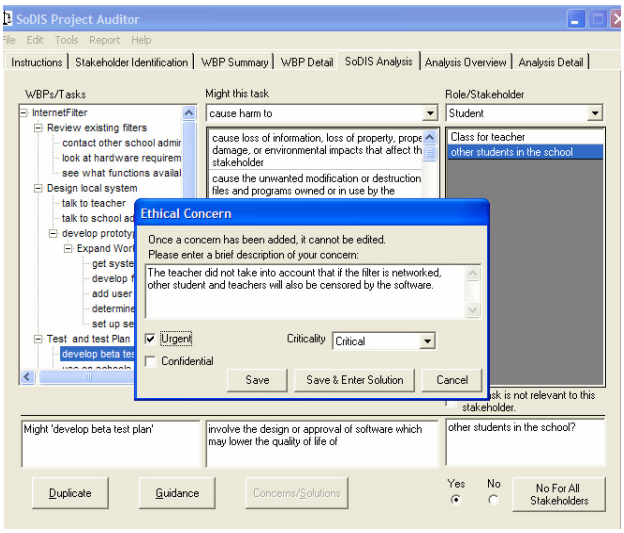

Figure 5: Concern screen

If they do have a proposed solution then they can enter it on the proposed solution screen (Figure 6).

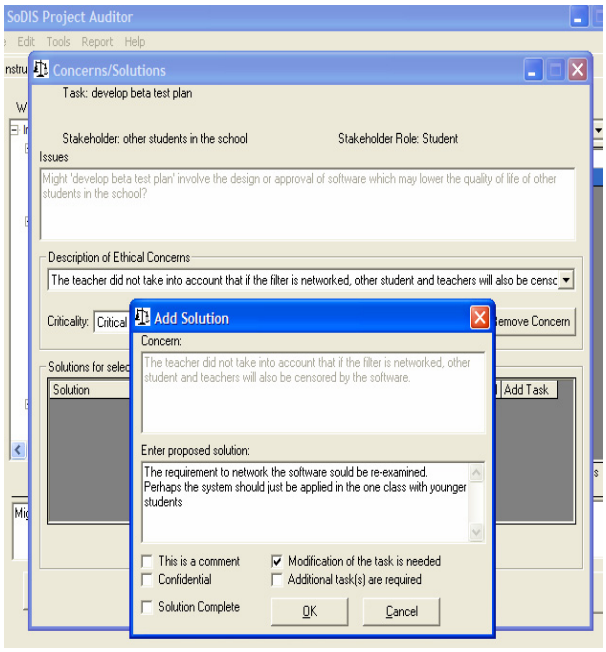

Figure 6: Proposed Solution Screen

The solution to the concern will require a change to an existing task(s), additional task(s), or both. The identified tasks need to be incorporated into the appropriate task lists to help eliminate or mitigate the risks identified. The early identification of these software modifications addresses qualitative issues, leads to a more coherent and ethically sensitive software product.

The prototype produces reports listing all issues identified, their assigned criticality and proposed solutions. This methodology can be applied to any phase of software development.

A complete SoDIS process 1) broadens the types of risks considered in software development by 2) more accurately identifying relevant project stakeholders. The utilization of the SoDIS process will reduce the probability of the types of errors identified by Farbey. The SoDIS should be part of any SDLC. Current research is being done on the place of SoDIS in agile development lifecycles.

The SoDIS process facilitates the expansion of software risk analysis to reduce software failures. Using this pre-audit process in tests in the UK and the USA facilitated the early identification of project risks. Using a SoDIS process in the classroom will make producing software of high quality and producing software that is ethically sensitive second nature for software developers. How can this process be used in teaching?

4 How to use it in Teaching

We can see several different opportunities both for you in your teaching and research and for us to help in improving the SoDIS process. Boud & Feletti [1997] suggest that "problem based learning is an approach to structuring the curriculum which involves confronting students with problems from practice which provide a stimulus for learning". We strongly recommend such an approach with the use of strategies such as case studies, scenarios and problem based learning designs which embed the SoDIS process and case tool in a practice context.

4.1 Teaching and reflective review by students:

The SoDIS Project Auditor (SPA) models three of the major phases in software development.

In the SPA guidance is provided to explain each type of question. Students could be asked to critique the guidance statement for the particular life cycle phase being discussed in class. They could research that area and write a one or two page revision of the SPA provided guidance with references. Thinking about what should be included in the guidance will help students better understand the issues. We would like to see the results of such exercises. Good ones would be used to improve our guidance descriptions (with appropriate credit given to instructor and student)

The SPA distinguishes different types of software projects. Students could be asked to find unique risk elements for a project type and develop questions that might help identify the presence of those risks within that type of project.

The SPA distinguishes several types of projects and assigns a group of default stakeholders to each project type. Students could be asked to review these lists for completeness.

The results of these activities would be useful to us improving the SoDIS process and this work would help your students to be reflective about the development process.

4.2 Software Development Projects:

1. Students could be asked to apply the SoDIS process to a sample case from the textbook you are using. They could record the new risks that are identified and describe how the project could have been done differently to achieve a better result.
2. Students could do a SoDIS analysis on the project plan for their software engineering projects.
3. Usability Testing: We would appreciate a general critical review of the tool and the process. For any problems that occur- a complete description of what keys were pressed and what data was entered would be helpful. This could be used to teach the students testing and test reporting procedures.
4. They could do a post-mortem on their project and identify where the major problems were and then review the SPA to see how it could be improved so it could help others anticipate the problems they encountered.
5. Have them develop their own question list and how it is related to the subject discussed in class. Risk issues, testing requirements
6. Good examples of these results could be distributed with the SoDIS to help others learn how to use it.

The potential for participation in improving the SoDIS case tool acts as an excellent motivator for the students.

There is clear evidence of its utility in teaching. Within the umbrella of the SoDIS SEPIA collaborative research programme (Clear 2003) the SoDIS process, and the SoDIS Project Auditor CASE tool, have been used in several different courses and institutions. In support of teaching the software process and risk and project management, it has been applied in a software engineering course at Monash University. It has been used in several capstone projects at Auckland University of Technology, where in one mid-project review of a project being undertaken for an industrial client it has proven helpful in identifying issues unaddressed by the project team, and helped bring the project back from the brink. An innovative use of the SoDIS Project Auditor has been reported by Koh (2003) who has used it for teaching the concepts of target audience analysis in a multimedia project. This project suggests that it may be valuable in both the early and later phases of projects of a variety of types, in which profiles of user communities is critical. Thus it may be a valuable tool in support of teaching aspects of a subject such as Human Computer Interaction. At Bay of Plenty Polytechnic the SoDIS process has been applied successfully in a project course and to the teaching of a course in ethics and professionalism with the spin off that students come to their later systems analysis and project management courses with a clear understanding of the notion of stakeholders and their needs. At Otago Polytechnic Smith and Mann (2003) have profiled the use of the SoDIS process as an auditing technique in assessing the ethicality of their own teaching. The process offers considerable scope for interesting educational activities, and can be included in many different courses to good effect. Research is continuing into effective ways for using the process in teaching, research and practice situations, and regular SoDIS SEPIA symposia are being scheduled (November 2003, and July 2004) in which participants in the research programme update their colleagues on progress, and share resources and ideas. The number of participants is growing and we are happy to see this continue.

5 Goals met using it in teaching

Several difficulties faced when showing our students key processes and techniques for software development have been noted above. Through use of the SoDIS process and its accompanying CASE tool the SoDIS Project Auditor, we are able to add considerably to student knowledge and awareness of the importance of context in their professional work, and the fact that their technical decisions impact many different stakeholders involved in the operation, use and results of use of the software they will develop.

The SoDIS process has proven useful in demonstrating to students the value of this extended form of risk assessment undertaken progressively throughout their projects. The process supports and improves their project management practices, ensures that they broadly consider impacts on stakeholders when developing their software, and demonstrates how risks unaddressed at early stages of the project compound as the project go on. This awareness expands to a growing comprehension of the burgeoning complexity of quantitative and qualitative risks, and a heightened understanding of the scope of requirements and the serious responsibilities resting with professionals in the field.

Use of the SoDIS process in teaching has also identified many innovative ways in which the process and the CASE tool have enhanced the learning experience and the curriculum across many different courses in the computing field. Many more opportunities are available for those willing to exploit the SoDIS process and the SPA in their own teaching.

6 Conclusion- an invitation

Research & Teaching:

The results of any of the items above provide valuable information for research on risk analysis in software development and could be formulated into conference papers and publications. We encourage you to consider using the SoDIS process and software in your own teaching to broaden your students learning, promulgate the process, and help develop a more informed group of graduates, IT professionals and developers for the software industry of the future. Copies of the SoDIS Project Auditor will be freely made available to support its use for educational purposes.

The authors wish to emphasise that this is an open research programme, and interested parties who share the aims of the SODIS SEPIA initiative (namely to improve the quality of software through the use of the SoDIS Process) are very welcome to join, by contacting any of the authors.

REFERENCES

Boud, D., Feletti, G., *The Challenges of Problem Based Learning,* Kogan Page, London, 1997

Clear, T., McHaney, R. Gotterbarn, D. (2003) *SoDIS SEPIA -Collaborative Partnerships in Software Engineering Research,* In *NACCQ Conference*, Vol. 1 (Ed, Mann, S.) NACCQ, Palmerston North, pp. 41- 48.

Davey, I. (2003) *Presentation to SoDIS Symposium on the Use of SoDIS In Teaching Ethics,* In *SoDIS Symposium,*(Eds, Mann, S. and Williamson, A.) NACCQ, Palmerston North.

Farbey B, Land F and Targett D (1993) *How to assess your IT investment*, Butterworth Heinemann.

Gotterbarn, D. (2001a) *Keynote: Understanding and Reducing Project Failure: The Ethics of Project Management,* In *14th Annual NACCQ Conference,* (Ed, Mann, S.) NACCQ, Napier, New Zealand, pp. 41-51.

Gotterbarn, D. (2001b) Reducing Software Failures using Software Development Impact Statements Journal of Information Management Systems December 2001

Koh, Donald (2003) *Using SoDIS for Target Audience Analysis: a Fresh Field Application,* In *NACCQ Conference*, Vol. 1 (Eds, Mann, S. and Williamson, A.) NACCQ, Palmerston North, pp. 103 - 107.

Rogerson, S., Gotterbarn. D. "The Ethics of Software Project Management", in *Ethics and Information Technology*, ed. Göran Collste, New Academic Publisher, Delhi, 1998

Smith, L, Mann, S. (2003) *Assessment of ethical processes in computing education: are we clean? How to tell?,* In *NACCQ Conference*, Vol. 1 (Eds, Mann, S. and Williamson, A.) NACCQ, Palmerston North, pp. 409 - 414.

Support

This research was partially funded by NSF Grant 9874684, and New Zealand funding has come from both the National Advisory Committee on Computing Qualifications (NACCQ) and the Auckland University of Technology Faculty of Business Contestable Research Fund.