Communications of the Association for Information Systems

Volume 15 Article 40

June 2005

Responsible Risk Assessment with Software Development: Creating the Software Development Impact Statement

Don Gotterbarn
East Tennessee State University, gotterba@etsu.edu

Simon Rogerson
De Montfort University, srog@dmu.ac.uk

Recommended Citation
Gotterbarn, Don and Rogerson, Simon (2005) "Responsible Risk Assessment with Software Development: Creating the Software Development Impact Statement," *Communications of the Association for Information Systems*: Vol. 15 , Article 40.
DOI: 10.17705/1CAIS.01540
Available at: https://aisel.aisnet.org/cais/vol15/iss1/40

Communications of the Association for Information Systems

RESPONSIBLE RISK ANALYSIS FOR SOFTWARE DEVELOPMENT: CREATING THE SOFTWARE DEVELOPMENT IMPACT STATEMENT

Don Gotterbarn
Computer and Information Sciences Department
East Tennessee State University
gotterba@etsu.edu

Simon Rogerson
Centre for Computing & Social Responsibility
De Montfort University

ABSTRACT

Limiting the focus of risk analysis to quantifiable factors and using a narrow understanding of the scope of a software project are major contributors to significant software failures. A Software Development Impact Statement (SoDIS) process is presented which extends the concept of software risk in three ways;

--it moves beyond the limited approach of schedule, budget, and function,
--it adds qualitative elements, and
--it recognizes project stakeholders beyond those considered in typical risk analysis.

As the types of risks increase, the range of stakeholders that need to be considered also expands. Using this expanded view of risk analysis reduced or eliminated the impacts of many previously undetected risks of software development. The SoDIS process and its software associate development tasks with relevant stakeholders through the application of structured questions.

This process was incorporated effectively into the software development life cycle and applied to software development projects in different domains on several continents. The successes of the SoDIS process provide strong evidence that a significant side-effect of narrowing project objectives is a root cause of IT project failures.

Keywords: Project Management, SoDIS, Risk, Ethics, Stakeholders

I. INTRODUCTION

Software developers continually evolve and refine techniques to mediate risk of failure in software development projects. They pay significant attention to risks which contribute to impeding a project; risks which contribute to missed schedules, budget overrun, and failure to meet the system's specified requirements. In risk analysis and mitigation literature the primary focus is on the project development vision defined in terms of budget and schedule overruns and not on satisfying the customer by meeting technical requirements. Henry [2004] defines project risk as

> *"an event, development, or state in a software project that causes damage, loss, or delay."*

Schwalbe [2004] also defines 'project risk' as

> *"…problems that might occur on the project and how they might impede project success."*

These common risks internal to the software development process, "intra-project risks" are managed and evaluated using quantifiable values. In a major KPMG study of runaway projects, failing to follow this risk analysis model was identified as the primary causes of project failure [KPMG, 1995]. Reliance on a high level generic risk analysis is inadequate or incomplete as a methodology. For example, 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 of the delivered project.

Why do the number of IT failures continue to be large even when risk analysis is applied during development? We argue that these failures are due in part to an institutionalised narrowing of the scope of a project's objectives and vision to development objectives. For example, consider the development of traffic control software to direct traffic approaching a multi-lane bridge into the least congested lanes to facilitate a maximum and continuous traffic flow across the bridge, especially in rush hours. From this description we would identify stakeholders in this software as including: vehicle drivers traversing the bridge, bridge maintenance people, and the city traffic authority. It is also straightforward to define success criteria for this software. They might include: the system works well in its context; it does not promote vehicle accidents; the project was delivered on time; the project was within budget; and the cost/benefit analysis was accurate showing that those developing the system could expect a reasonable return on investment. The system met all of these conditions and yet it was judged a failure. Why?

The system needs to manage large amounts of traffic moving through 20 lanes. Cars go over the bridge at two levels. The computer must make continuous interactive rapid and accurate processing decisions about such quantities as lane capacity, average speed of the lane, stopped lanes, taking lanes out of use, changing directions of lanes to account for rush-hour flows. The system was installed and worked well until the system was required to manage constant heavy traffic loads for 8 hours during an emergency nuclear disaster evacuation exercise. In the eighth hour the software changed lane directions for lanes already filled with cars and the misdirection and accidents clogged the bridge for almost 20 hours. The crystal clock used for the timing of these decisions would gradually go out of synchronization with 7 or more hours of continuous use. The developer was fully aware of this problem. To meet the problem, the developer specified in the user manual that the software should be briefly stopped and restarted after 6 hours of continuous heavy traffic loads. This action would reset the clock and no problem would be encountered.

The vision, objectives, and stakeholders were all considerably narrowed when the project became a software project. The order of the success conditions listed above was also reversed. The primary goals were to deliver the system on time, within budget, and satisfying the customer. The focus of the risk analysis and mitigation narrowed to those many issues which impact these goals negatively and risks that would derail the project's development. This narrowing of focus to development risks is canonised in many information systems and software development

textbooks and risk management articles [e.g., Henry 2004, Schach 2003, Schwalbe 2004] and documented in official development standards [CMM, PMBOK, ISO929]. To meet schedule and budget constraints, the bridge software developers opted merely to place a warning in the user manual rather than provide a software solution. The choice to focus on high level intra-project risks and stakeholders is related directly to the software development.

One of the typical ways to address risks is to focus on the quantifiable risks related only to those directly involved in the development of the software and overlook identifiable qualitative risks. For example, Watts Humphrey, a fellow of Carnegie Mellon's Software Engineering Institute, defines good software as "usable, reliable, defect free, cost effective, and maintainable" [Humphrey 1986]. Humphrey's focus is on software characteristics: how many remaining defects; how long the software will continue to run; and the simplicity of the design which is a way to quantify maintainability. He does not consider the potential negative impacts of the software.

Extended action research and Delphi studies by Lyytinen [1987], Keil [1988] and Schmidt [2001] categorizing and extending traditional software risk confirmed the failure and consequences of this narrow focus on risk. This kind of limited quantitative approach contributed to numerous software failures, some of which received very public notice. 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 function and budget to the exclusion of others, led to developing an interface that was inadequate for use by the sailors (stakeholders in this software) in a combat situation. Fortunately, not all software failure impacts are of this magnitude, nevertheless the problem is significant. Mackenzie [2001] confirms the view that a narrow focus frequently leads to software "that works brilliantly but doesn't fulfil the need".

Research results indicate restricting the scope of types of risk factors considered is inadequate for effective risk management. Boehm [1989] and others argue that risks must be identified before they can be addressed. Schmidt et al. [2001] catalogued and categorised 53 risk factors. A mechanism is needed to identify additional potential risks that, Schmidt et al. [2001] did not identify in the 53 project risk factors they catalogued.

The generic quantifiable approaches to risk focus on 'complexity' in terms of the number of lines of code or number of function points. Often systems are evaluated in terms of the number of faults per 1000 lines of code rather than the side-effects these faults may have on system users or those affected by the system. These are interesting numbers but they mislead developers in their specificity. This numerical approach is now canonical in the software engineering methodology literature in approaches like the Personal Software Process [Humphrey, 1996], the Team Software Process [Humphrey, 1999] and the earlier Capability Maturity Model [SEI, 1995].

This emphasis on quantitative measures is seen in a process improvement presentation, by Gabriel Hoffman [Hoffman 2003] of Northrup Grumman who described the quality of their software development process. He included as measures of quality: the number of hours saved in production, the number of defects per thousand lines of code, low schedule variance, an improved design code *ratio*, defect density, code size, and cost variance.

Software development's shift of project vision contributed to the narrowing of focus on specific types of risks, an emphasis on quantifiable risk almost to the exclusion of qualitative risk. This emphasis on quantitative risk contributes to an underestimating or ignoring of the need to consider risks to extra-project stakeholders in the development of the software. Schmidt points out that the "[f]ailure to identify all stakeholders: Tunnel vision leads project management to ignore some of the key stakeholders in the project, effecting requirements, and implementation, etc." [Schmidt et al. 2001 p 15] Project risk analysis must be expanded beyond the traditional risk analysis to include a broader scope of risks and stakeholders. The need for a formal mechanism to expand scope is evidenced by the discovery of the additional difficulty.

> *"that managers are biased in their attention to particular risk factors [schedule, budget, function] at the expense of other factors* [Schmidt et al. 2001 p.26]."

Those responsible for software development need to be fully informed about all aspects of risk if they are to increase the likelihood of success rather than failure. Indeed, Keil et al. [2000] found that,

> *"risk perception appears to have a much more significant influence on decision-making than does risk propensity. This result is significant because it implies that decision-making can be modified through manipulation of risk perception, in spite of any individual differences that may exist in terms of risk propensity. Thus, it may be possible to design risk assessment instruments and other interventions that reduce the incidence of project failure by altering managers' risk perceptions."*

In this paper, we describe a process, the Software Development Impact Statement, abbreviated SoDIS, that improves and expands risk perception. We believe that enhanced risk perception should reduce the dangers of a narrow focus on quantitative risks.

II. ADDRESSING THE RISK ANALYSIS PROBLEMS

The problems of risk analysis can be addressed in several ways including:

- expanding the list of generic risks,
- maintaining focus on the broader project goals, and
- extending the list of considered project stakeholders.

RESEARCH BY PROJECT TYPE

Generic risk analysis limits the developer's perceptions. Failure to identify, understand, and address the risks associated with different types of software projects is a key contributory factor to project failure by constraining the developer's perceptions of the real project risks. It is commonplace to define project types by the nature of the software being developed. For example, Jones [2000] classifies software development project types as systems software, commercial software, internal information systems, outsourced software, military software and end-user software. Software development projects differ in a number of significant ways. For example, real-time military applications differ from commercial batch applications in their technical risks. Software development risks differ by types of software projects. The systems delivered. also involve different types of stakeholders and risks. Recent research documents the need to extend standard project risk analysis to include analysis of the risks created by the delivered system. For example, Schmidt et al. [2001] studied and correlated types common software project risks factors and the attention given to them. After looking at the business-related stakeholder and theories which limit the extension of stakeholders considered to users of the software, Schmidt concludes that these extended risk factors remain "largely unexplored areas in software project risk management."

For a project development to succeed, risk resolution should consider:

- the delivered project type, consisting of sector and application;
- all stakeholders' opinions; and
- the different stakeholder expectations about how to judge a project as a success or a failure.

Even for the simplest of projects, with a small development team working on software with low complexity and limited functionality, responsible risk analysis requires categorisation and description of the delivered project, and the associated direct and latent stakeholders.

For example, in a recent case in New Zealand, a software developer was asked to develop an Internet filter which would only allow a browser to access web sites which were on an approved list. Based on this description a developer addresses the generic risks of schedule delay, incomplete functionality, and cost estimation. Software was developed which allowed one administrator to enter and remove web sites from the list. The list filtered all Internet access. It was delivered on time and under budget. At installation the developer learned that the filter was to be installed in a school that was going to network all of its computers. If we merely consider the functionality then one set of risks in development would be addressed. The contextualization of the school setting identifies delivered project risks and changes the way in which the software should be developed to mitigate these newly identified risks. The project is now constrained by needs of administrators, teachers, and students who will be using the network and the Internet and those who may find access to their sites prohibited.

These extra-project risks involve stakeholders beyond the project team and the customer. It is the failure to consider these 'extra-project' risks and 'extra-project' stakeholders which make this project a failed one. The inattention to these risks and stakeholders contribute to estimates from the Product Development and Management Association that the failure rate of newly launched software products is approximately 59% [Cooper 2001].

CATEGORISING PROJECTS TO AID RISK IDENTIFICATION

How can we best categorize a project to promote this improved perception of risk? Among the various answers offered to this question are:

- The size and complexity of the software and thus the project will impact the types of control and monitoring tools used by the project manager. Current thinking using the nature of the project appears limited to aspects of the development process and development environment.

- Simply categorise projects by size of the code or duration in order to guide their risk management approach. For example the Prestwood Software Development Process [Prestwood Software, 2002] defines software, using a look-up table, as 'small', 'medium', or 'large' and this designation is used to determine the number of iterations of the development cycle.

- Categorise by the technological aspects. For example Shenar et al. [1996] use four levels of technological uncertainty to ascertain the way to develop software.

These approaches do not shed much light on the risks of the delivered Internet filter discussed earlier. The problem we identify with these approaches is that they are inward looking, focusing simply on the obvious within the narrowly confined boundaries of the development of the software. The inattention to potential side-effects – extra-project impacts - leaves the system development vulnerable to unforeseen problems and risks which can radically hamper progress, cause flawed implementation, and lead to eventual total failure of the development or failure of the delivered system. Such situations are easy to imagine in the Internet example. The risk of limiting a teacher's Internet research by applying student access constraints to the networked filter would be missed if the teacher stakeholders were not considered. The system could be delivered on time, within budget and meet the filtering functionality but would be a failure. It would be one of those projects that are delivered but never used. This project failure was caused by the same narrow focus on stakeholders directly related to development that contributed to the Aegis disaster (Section I).

Risk analysis should also be outward looking and take into account the overall environment within which the software will be used and the target application area. This contextualisation directs the focus to relevant stakeholders and colours the way in which more traditional aspects such as size and technological complexity are considered. In this way, everyone involved in the software development or affected by its delivery are catered for and, by implication, the process of

development as well as the outcome (the software) is properly reviewed. The lack of a good risk perspective by contextualisation of the product is a widespread deficiency in software development. For example, Aladwani [2002] found it was essential to understand the context of the project to increase the likelihood of project success in developing countries, yet such consideration was usually missing [cf. Gotterbarn and Clear, 2004]. Given a project's development process is simply a means to an end and not the end itself, then the contextualisation of a project deliverable should focus on the impact space of the software rather than just an introspective focus on the development process. Therefore, in this sense, contextualisation involves three main dimensions:

- the sector within which the software will be used;
- the type of application that is to be addressed; and
- the application's surrounding circumstances.

These dimensions are important in adequate risk and stakeholder identification. In the Internet filter project, the context of a school changes the types of risk that need to be addressed in this software development. Moving from the context of a single classroom to the installation of one filter on a network for the entire school involves new risks including technical issues, privacy issues, and restrictions on illegitimate access. In the bridge example the failure to modify the system to avoid the errors caused by multiple hours of use during emergency nuclear disaster evacuation exercises led to not being able to use the system in its time of greatest need. Clearly all of the elements of a product's context; sector, application type, and circumstance, are required for an adequate project categorization and for effective risk analysis. The results of this analysis of product risk by product contextualization must be added to development risk analysis for a complete risk analysis for system development.

III. IDENTIFYING STAKEHOLDERS

In addition to inadequate project categorisation[1], misidentified or unidentified stakeholders are a major contributory factor to the ineffectiveness of current risk analysis methods. Some researchers try to develop generic methods of stakeholder identification that they believe can be domain-independent. For example, Sharp [1999] and Henry [2004] identify methods based on either direct stakeholder interaction with the system or financial involvement with the system. Stakeholders are "people with a direct internal involvement or investment in a software project" [Henry, 2004]. This common approach ignores the special circumstances generated by the nature of the product delivered. A patient waiting to be identified by software as a heart transplant recipient would not be considered a stakeholder during development. Other even more limited views are that the only critical stakeholders are the project team members [Spafford 2002]. Such views of who are the relevant stakeholders are clearly too limited. The need to expand the stakeholder group is supported by Keil et al. [2002].

> *"Incorporation of the user perspective on risk is significant because focusing solely on project managers' perceptions may result in some risk factors receiving a lower level of attention than they might deserve. To mitigate project risk, it is necessary to consider all risk factors judged to be important by both groups and then reconcile differences through dialogue, deliberation and communication."* Keil et al.[2002]

As we have seen, a fully responsible risk analysis needs to go beyond even Keil et al.'s inclusion of a user as an extra-project stakeholder.

---

[1] Categorisation is discussed under "Categorising Projects to Aid Risk Identification" in Section II and in the material that follows.

The concept of "stakeholder" is used in many different ways. One extreme talks of "stakeholder" as "participants in corporate affairs" [Ackoff' 1974]. The stakeholder should have some financial stake in the corporation. While Lyytinen and Hirschheim [1986] extend the realm of "stakeholder" further to include those who expectations go beyond the requirements since only a fraction of a stakeholder's concerns are usually formulated in the requirements. Lyytinen and Hirschheim use this definition of "stakeholder" to argue that many IS failures actually met the requirements, but were considered failures because some other vital concerns were not met. Our use of "stakeholder" is closest to Willcocks and Mason [1987] who define the stakeholders of a computer system as the

> *"people who will be affected in a significant way by, or have material interests in the nature and running of the new computerized system" (p.79).*

These various concepts of "stakeholder" each provide associated techniques, some purely quantitative, to aid in identifying project-relevant stakeholders. A qualitative approach to stakeholder identification is suggested in Section VI, which opts for the more general concept of stakeholder. Successful project management needs to consider the people impacted by the system.

In the Internet filter example, the stakeholders initially consisted of the teacher requesting the filter and the developer; when the filter was placed on classroom computers the stakeholders expanded to the students in the class. Then, because the computer was networked, the stakeholders changed again. The stakeholders for the Internet application changed as it was placed in the education sector and changed again with the changes in circumstance. In part these changes are related to the contextualization of the product. Developing and using an expanded standard checklist methodology facilitates identifying stakeholders directly and indirectly associated with the project deliverables. A preliminary default lists of stakeholders associated with and affected by particular project types ensures their consideration during the risk analysis. The complete contextualization of a system and identification of relevant stakeholders is only part of a satisfactory risk analysis. The risk analysis problems of narrowing the risk focus to generic risks is compounded by limiting the way such risks are analysed.

IV. QUANTITATIVE VERSUS QUALITATIVE APPROACHES TO RISK

Quantifiable risk analysis is critical for good judgement in software development. A quantitative approach to risk relies on developing metrics that can be used to describe the risks in terms of money lost, days over schedule, numbers of functionalities not met. These quantities can be measured periodically to ascertain the existence and severity of risk in terms of Risk Exposure and Risk Leverage [Boehm, 1989]. The two problems with this emphasis on quantification are:

1. the emphasis on quantifiable risk to the exclusion of qualitative risk; and
2. how this emphasis changes the risk perception of the developer by only admitting those quantitative risks which result in quantifiable intra-project impacts.

This approach impacts risk perception because it limits the concept of "software failure". "Software failure" is not simply an issue of schedule, budget and reliability. As Kuruppuarachchi et al. [2002] point out,

> *"the effective management of changes in a sociological context, not only in economic and technological terms, is a prime requirement for success. The project's success is determined by customer acceptance of the project rather than the factors such as budget, timeliness or technological sophistication. Software has been developed which, although meeting stated requirements, has significant negative impacts on the circumstances, experiences, behaviour, livelihood, or daily routine of others."*

In the Internet filter project, the system promotes censorship by omission. The system requires constant monitoring by the school's designated Internet censor. The system limits student preparation for later courses. In general these types of qualitative issues with software are recognized but treated inadequately by information systems professionals.

This problem of addressing both quantitative and qualitative risk is a global issue. For example, based on empirical research, Dey [2002] recommends that best practice for Caribbean organisations must rely upon balanced risk analysis based upon

- *"aligning the project goals with the strategic intentions of the organisation";*
- *"appropriate requirement analysis with the involvement of project stakeholders"; and*
- *"equal emphasis on all aspects of analysis (market and demand analysis, technical analysis, financial analysis, economic analysis, and impact assessment)".*

This clear inclusion of qualitative risk analysis helps to address the Caribbean governments'

> *"need to strengthen the planning and development framework in the public sector, as the basis for improving the delivery capacity and economic performance of development projects."*

In the Internet filter project, risks like over-constraining Internet access and security of the "approved web sites list" are not quantifiable but can be categorised by their impact on the project. Sometimes these qualitative risks are converted into financial impact on the project. However this limited approach to qualitative risk is based on the narrow views of stakeholder and type of project challenged above. When a project's deliverable is properly contextualized and the relevant stakeholders identified, then the set of qualitative risks is extended. In most cases these risks cannot reasonably be quantified but only categorized by their side-effects on extra-project stakeholders as 'critical', 'significant' and 'minor'. This categorisation provides an understanding of unique types of risks thus facilitating the discovery of appropriate solutions for heretofore unidentified risks.

Information Systems professionals frequently fail to give appropriate weight to one of the approaches to risk analysis. Because of the focus on ROI and quantifiable issues, even when they do qualitative analysis they limit it with quantitative constraints. It must be recognised that qualitative risk analysis and quantitative risk analysis are complementary and both are necessary.

Some of the risks missed by quantifiable risk analysis were documented in a detailed empirical analysis of two failed systems cases by Dalcher and Tully [2002]. They confirm that software failure is the result of 'multiple, complex and interrelated' causes. Two of the failure causes they identify are:

- 'deafness to alerts' which concerns the lack of interest by senior management, project management and technical developers making project-related decisions to warnings given by a variety sources often beyond the traditional stakeholders; and
- 'groupthink' which is an active resistance to any outside influences that might threaten the accepted norm mindset.

Together these causes justify a new approach to how risk analysis is undertaken and by whom.

> *"It is essential that … stakeholder groups are integrated into the system process, with each having an effective voice so as to be able to express both their needs and their knowledge."* [Dalcher and Tully, 2002]

It is recognised that stakeholder involvement, particularly by indirect stakeholders, can be difficult to realise but this must not deter developers from striving to ensure all stakeholders' interests are properly represented during risk analysis. Unfortunately, at times some stakeholder's interests

can only be included by developer's speculating about external stakeholder needs. Pouloudi [1997] provides mechanisms to facilitate the identification of stakeholder needs.

V. TOWARD EXPANDED RISK ANALYSIS

We developed a risk identification process to complement existing quantitative risk analysis methods and reduce the number of software failures. The risk analysis method is called a "Software Development Impact Statement" (SoDIS) [Gotterbarn, 2002]. The SoDIS process expands existing software development risk analysis methods by developers explicitly addressing a range of qualitative questions about the impacts of their development from a stakeholder perspective. SoDIS overcomes the limitations described in Section IV.

The SoDIS process was tested on software development in organisations with different location, size, function, scope, development methodology, and technology level; from small projects in consulting companies to projects as large as the United Kingdom's scheme for Electronic Voting. In one case, using blind tests, risks were identified which could saved the company $250,000 USD. Applying the SoDIS method to system development documents from known software failures, novices were able to anticipate the potential risks that were realized in the actual project. Just as quantitative risk analysis can be applied at every stage of software development, SoDIS was tested successfully against every phase of development. [Gotterbarn, Clear, and Kwan, 2004].

The SoDIS process belongs to the family of issues-oriented approaches used in software systems development. Early approaches, like Hirsheim and Klein [1989] proposed to expand the scope of consideration within a system development project and add "quality of the work life" (which encompasses for example working conditions and progression opportunities) and ethical concerns. This inclusion of "quality of the work life" does not fully address the complete set of impacted stakeholders in information systems development. This family of approaches also includes Soft Systems Methodology (SSM) which can address "improvements to problem situations and lessons that can be learned in the problem solving process" [Checkland and Howell, 1998; Jayaratna, 1994]. One approach to SSM takes a holistic perspective of analysis. Another member is ETHICS which is concerned with "the design process and in encouraging the participation of those organisational members whose lives may be affected by the design" [Mumford 1996, Jayaratna, 1994]. ETHICS takes a restricted stakeholder perspective of design.

Keil et al. [2002] argue that stakeholders, in particular developers and users, harbour different perceptions regarding potential project risks. They showed empirically that, by understanding and taking these differences into account, the chances of successful software delivery increases. SoDIS is different from the Keil et al. approach in that it takes a comprehensive stakeholder perspective of the whole development cycle because it considers each task within the structured plan of the project. A SoDIS risk analysis can be applied to any work product such as a work breakdown structure in a system's development. It can function as a review or preliminary audit of any development milestone. In this way it can seamlessly fit within a software engineering approach to development [Gotterbarn, 2004] whereas both SSM and ETHICS cannot inasmuch as their roots are in interpretative analysis.

VI. THE SoDIS SUMMARY

The Software Development Impact Statement (SoDIS) process is a modification of the environmental impact statement process. A SoDIS, like an engineering environmental impact statement, is used to identify potential negative impacts of a proposed project and specify actions that will mediate these anticipated impacts.

A SoDIS risk inspection process [Gotterbarn, Clear, and Kwan, 2004] includes steps to complete the project contextualization and maintain the view of the project scope. Inspection is followed by a detailed SoDIS audit process which can be applied at various points in information systems

development. The SoDIS audit process expands existing software development risk analysis methods by helping developers identify the appropriate set of project stakeholders or entities and by examining the impact of the tasks of software development on each of the stakeholders.

Although similar on an abstract level, in practice software development projects vary on several dimensions such as

- size
- complexity/uncertainty
- the context,
- application type
- circumstances of the project

Variations on these orthogonal axes need to be acknowledged because they change the stakeholders who need to be considered in a complete risk analysis. For example, imagine how a simple change of context in the Internet filter project from a school to a penal institution would significantly alter the risk analysis or how a change of context from directing bridge traffic at fee lanes to directing Patriot missiles would alter the risk analysis.

The goal of the SoDIS audit process is to identify significant ways in which the completion of individual tasks, that collectively constitute the project, may negatively affect intra-project and extra-project stakeholders. It identifies additional project tasks that may be needed to prevent any anticipated problems and identifies changes that may be needed in some tasks to prevent anticipated problems.

As shown in Figure 1, the SoDIS process consists of four stages:

1. Identifying the project type together with immediate and extended stakeholders in a project,
2. Identifying the tasks in a particular phase of a software development project,

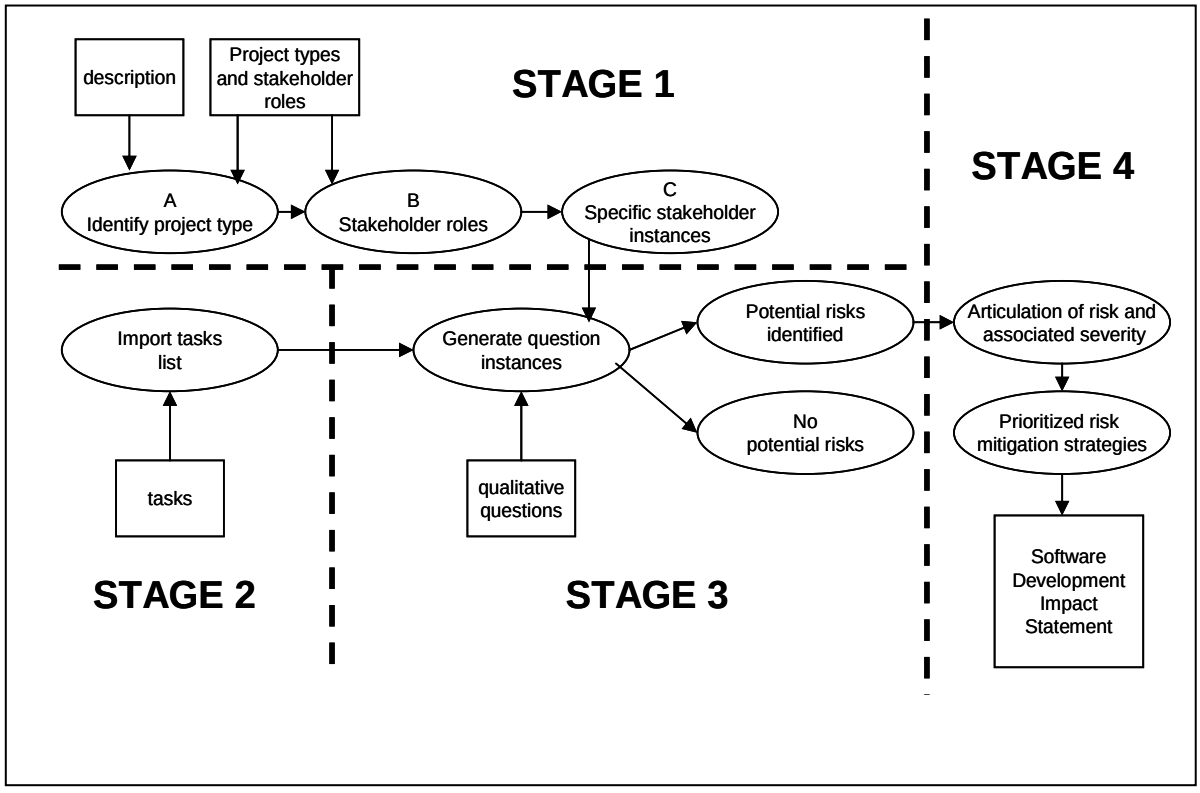

Figure 1. The SoDIS Process

3. associating 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 stating the concern and the severity of the risk to the project and the stakeholders, and recording a possible risk mitigation or risk avoidance strategy.

The resulting document, which complements a quantitative analysis, is a Software Development Impact Statement (SoDIS) which identifies all types of potential qualitative risks for all tasks and project stakeholders.

This process can be done both bottom up and top down. The SoDIS process can be applied at any level of a hierarchy of tasks. As new risks are identified, any stage of the SoDIS process can be revisited for any task level. This flexibility is significantly different from the environmental impact model which makes a single pass at the project concept at a very high level and leaves risks that can only be discovered with more information later in the process unaddressed.

The SoDIS is the missing element in current risk analysis which primarily focuses on some of the quantitative intra-project relationships between selected tasks and selected stakeholders that constitute a software development project. A responsible professional risk analysis examines both the quantitative and qualitative associations between tasks and project internal and extended stakeholders. To leave out either the quantitative or the qualitative analysis results in unidentified and, worse yet, unaddressed risks and project failures. We will illustrate this claim with the Internet filter project discussed above.

STAGE 1 - DEFINE PROJECT AND IDENTIFY STAKEHOLDERS

The first stage of the SoDIS process involves:

- categorizing the project type,
- identifying its default project stakeholders and
- expanding the default stakeholder list based on the unique attributes of the particular instance of the project type.

The specific project information is first developed in the early stages of the SoDIS inspection process.

Project Type Identification (1A)

To organise project types in a way that helps to identify their unique risks we chose two of the three orthogonal models: sector and application.

Sector - Given the diversity of software development projects, it is reasonable to ask, “Exactly what sectors should be identified?” Many detailed socioeconomic classification systems are in use, for example the Standard Industrial Classification (SIC) in the UK, the North American Industry Classification System (NAICS), and the World Bank groupings. The software development risks to address differ in each of the socioeconomic groups but also include some common risks. Using this idea of classification, several sectors appear to incur unique types of associated software development risks. These sectors are government, education, medicine and military, together with those systems developed for key internal use across other sectors such as real-time, internal or system projects. The risks and standards of quality of the Internet filtering in the education sector differ from those of a military project although both could also be viewed as internal projects. Given the evolving nature of the contexts of Information and Communication Technologies , new sectors will be identified.

Application - The application area is a second form of classification which helps to identify system development risk. For example Turban et al. [1996] suggest system classification can be done according to organisational level (e.g. departmental), major functional area (e.g. manufacturing), support provided (e.g., transaction processing) or system architecture (e.g.

distributed system). These types of classification can be used to identify types of applications that might exhibit unique risks. These applications include real time systems, scientific systems, and system software. The Internet filtering project would exhibit one set of characteristics when used by one teacher in one class room and a very different set of characteristics when it is being networked to every computer in the school, including computers used by all the other teachers.

Circumstance - The third element of delivered project's contextualization is identification of circumstance. Project circumstance cannot be specified because of its diversity. The circumstance of the project is used as a key to adequate stakeholder identification. Therefore, the SoDIS comes at the circumstance question indirectly as follows.

In addition to identifying unique project circumstances during the SoDIS inspection, the SoDIS audit portion of the inspection uses three questions which force an analyst to understand the context to answer these three questions:

1. Whose behaviour/work process will be affected by the development and delivery of this system or project?
2. Whose circumstance/job will be affected by the development and delivery of this system or project?
3. Whose experiences will be affected by the development and delivery of this system or project?

Answering these questions requires stories or scenarios about the software's various contexts. Answering these questions identifies the stakeholders related to the project in a given circumstance.

The three elements of sector, application, and circumstance complete the contextualization of the project. The relations among these elements can be seen in Figure 2 from a prototype software tool which implements the SoDIS process.

The *Type* drop-down menu lists applications and sectors from which to choose. Once chosen, Type determines the list in the *Role* drop-down menu for the stakeholder list. By answering the three circumstances questions the stakeholders can be added in the *Name* field for each *Role*.

Project Categorization- Judgement

In addressing software development impact the intention is to select the dominant project type for the sector group or application area. The aim is to focus risk analysis to extend the project manager's risk perspective beyond intra-project stakeholders to several extra-project stakeholder groups. Thus, those involved in the risk analysis must choose the dominant project type from a shortlist of Government, Education, Medicine and Military, Internal, Real Time, Scientific and System although the project may cover more than one type. The Internet filter example is both an educational project and an internal project. Considering the project primarily as an educational project identifies several extra-project stakeholders.

Stakeholder Roles (1B)

The stakeholder roles typically associated with any project include the developer and the customer, but we have already seen that more stakeholder types need to be considered even in projects as simple as the Internet filter. A survey by Farbey, Land and Targett [1993] of successful and failed projects found a minimal generic set of stakeholder roles need to be examined including: Customer, Developer, Project Team (including SQA), User, Vendor (Publisher), and Community. The contextualisation of a project implies slightly different clusters of stakeholder roles for each project type. For example, the development of educational software includes students and educators as stakeholders; researchers are stakeholders in scientific projects, and the development of aerospace navigational software involves astronauts and pilots as stakeholders. Ignoring these project-specific stakeholders yields incomplete risk analysis. During Stage 1B, the generic and specific stakeholder roles are identified and used in Stage 1C.

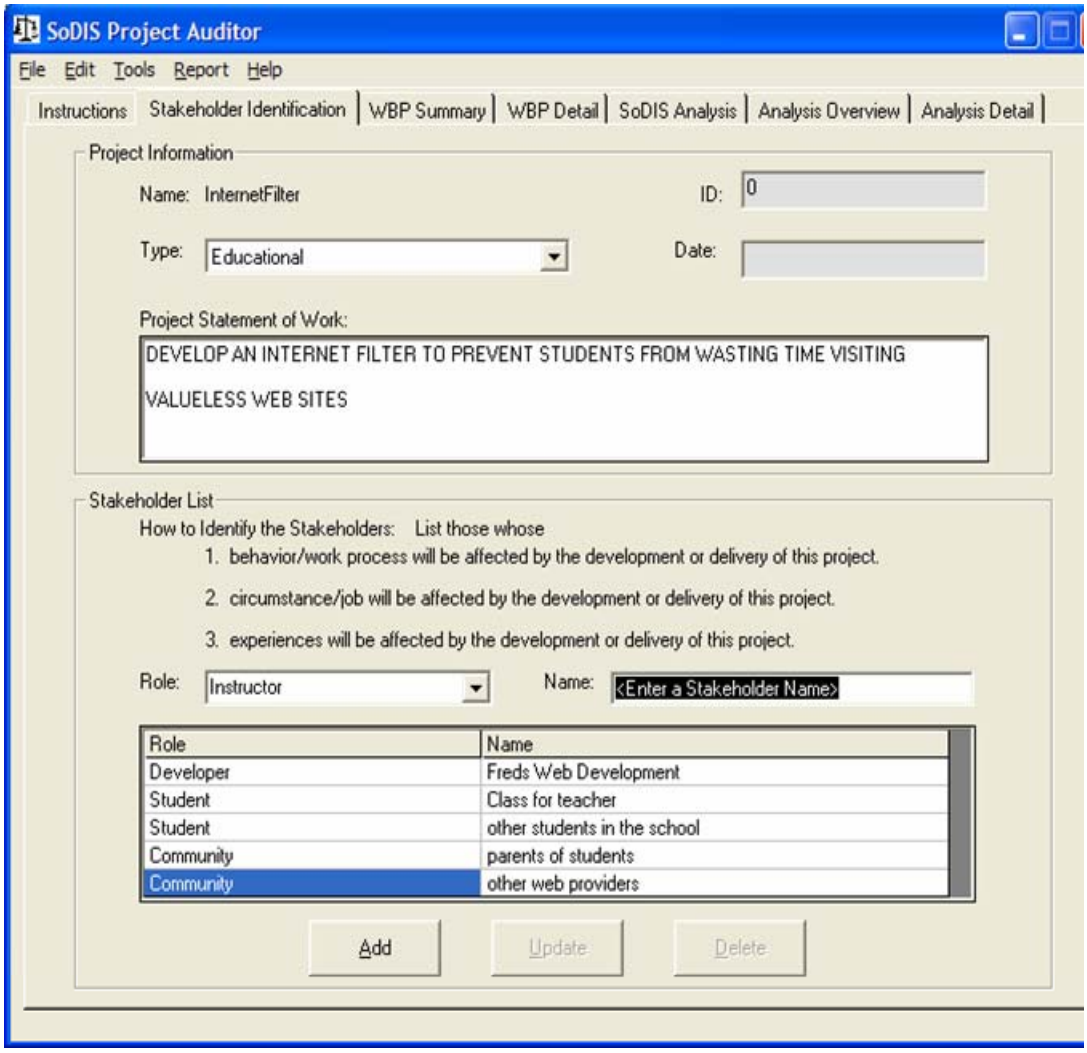

Figure 2 Contextualization of the Project

In the Internet filter project the stakeholders would include all the teachers whose use of the Internet for class preparation is inappropriately restricted by the specified filter.

Stakeholder Instances (1C)

The circumstances identify a particular instance of this project type. To complete the contextualization of the delivered project requires that specific relevant stakeholders must be identified. In traditional risk assessment, the focus remains on those who are considered key stakeholders. Other stakeholders and the ethical responsibilities owed to them by software developers and project managers are usually given little attention. Lyytinen [1987], Keil [1998], Raponnen [2000], and Schmidt [2001], also found Boehm's [1989] 10 risk factors incomplete and assembled a fuller lists of software project risks. But, even in these studies, the User is the only stakeholder considered outside of the original development team and business stakeholders.

Many project failures are caused by limiting the range of possible system stakeholders to just the software developer and the customer, thereby limiting the understanding of the scope of the project. In their research on failed projects, Land and Targett [1993] found that, with the exception of vendors, all stakeholders involved in evaluation were internal to the organisations involved in

the development. These people tend to be the only stakeholders originally identified among traditional project goals. A limited scope of consideration leads to the development of systems that result in surprising negative effects because the needs of relevant system stakeholders are not considered.

Those development models that address risks typically do so in this narrow sense. For example, even Boehm's [1998] "spiral lifecycle", which specifically addresses risk, limits stakeholder consideration to those people that impact the project. He also limits the risk to the three major traditional risks: budget, function, and schedule. Similarly, cost benefit analysis undertaken at the beginning of most projects only takes into account the interests of those involved in the analysis. The SoDIS process differs from other stakeholder analysis methods because it identifies a broad range of potential project stakeholders who need to be considered during project development.. For example, stakeholders in the Internet filter project include: 'other teachers in the school', 'the network administrator', 'the person who maintained the list of acceptable websites', and 'the owners of unlisted web sites.' If any one of these stakeholders was not considered then the software would be inadequate and would require significant change after installation.

STAGE 2 GENERATE THE TASK LIST

In Stage 2 of the SoDIS process, a list of tasks is generated. This list could simply be a work breakdown structure from a standard project management package. 'Tasks' is used as a generic term to identify the elements requiring our attention in each phase. These elements might be activities in a project plan or lists of functions from a requirements specification. SoDIS has been used to analyze different phases of software development. For example, in New Zealand it was used on a "task list" consisting of site maps and story-boards during the development of an e-commerce system.

The close relation to the task structure provides a link to a standard engineering approach to software development, emphasizing modularity and decomposition of the work into a hierarchy of tasks. The examination of the task list sometimes generates the awareness of new stakeholders and at that point they should be added to the stakeholder list.

STAGE 3 IDENTIFYING RISKS

In Stage 3 of the SoDIS process, potential qualitative risks are identified by associating tasks with stakeholders through pre-defined structured questions. A set of permanent questions for the process was derived from international codes of practice and conduct. These questions articulate qualitative issues common to all software development projects, such as

> *"Might <task> cause loss of information, loss of property, property damage … that impacts the <stakeholder>?"*

These questions (the qualitative glue holding the task and stakeholder together) ought to be addressed in any software project. As new tasks or stakeholders are added iteratively the risks can be analyzed similarly. New project specific questions can be added to the set of permanent questions. Answering these questions identifies potential qualitative risk. The prototype screen in Figure 3 shows a generated question about the Internet filter project.

The SoDIS analyst is asked to answer the question formed at the bottom of the screen about the potential impacts of the task on the stakeholder. Decision making and problem solving in software development go far beyond the structured realms of traditional software engineering. Failure is likely to be caused by a lack of recognition of and/or an ineffective approach to risks which are non-routine and unfamiliar or novel. Some of these risks require a qualitative approach to risks demanding intuition and judgement in their resolution. Concerns are initially recorded without pausing to assign a specific quantitative probability to the potential occurrence of the risk. In many cases the types of risk identified are not amenable to a precise probability assignment.

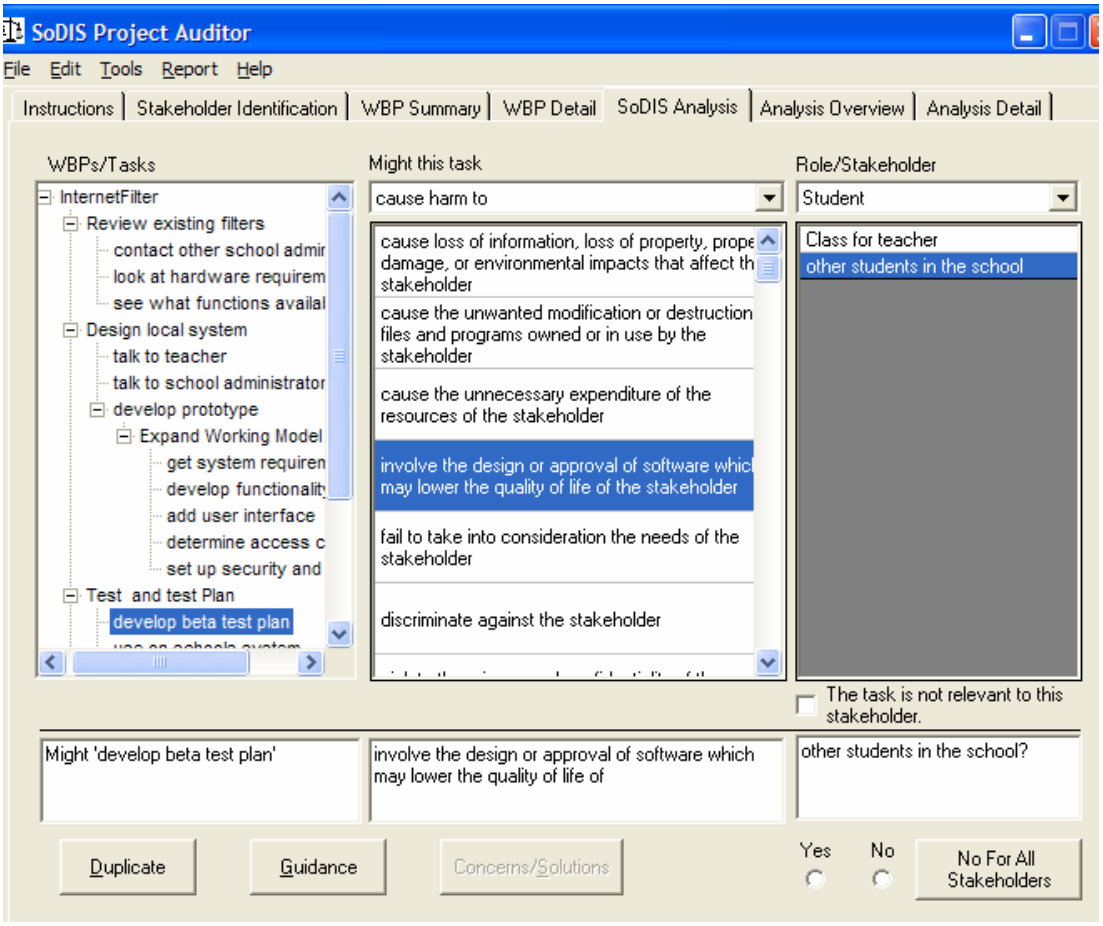

Figure 3 Example of a Generated Question

To address qualitative issues such as ‘deafness to alerts’ and ‘groupthink’ (mentioned earlier) a group decision approach is required which brings to bear a rich variety of values and beliefs of its diverse members. Group decisions promote collaborative creativity which facilitates successful implementation (Laudon and Laudon, 2005).

It is these concepts of extended qualitative risk analysis and group decision making on which the SoDIS process is founded. The SoDIS process uses active decision support in that it provides “mechanisms for the analysis and manipulation of information to provide greater insight into the decision situation and the associated options” [Fidler and Rogerson, 1996]. The SoDIS process supports standard group decision making, problem identification, and problem resolution techniques such as the Nominal Group Techniques, Delphi Techniques, and Brainstorming.

Group decision making in SoDIS works [Gotterbarn, Clear, and Kwan, 2004]. For example, with minimal instruction, a group of Australian Defence Department personnel, a group of New Zealand software engineers, a group of Polish university computer science tutors, and a group of UK computer science postgraduates, were all able to use the SoDIS process to identify potential risks in real world projects which were not identified by other means.

Risk analysis is often unsystematic, composed of recently noticed hotspots. This lack of a systems approach leaves open the possibility that later in the process a developer does not know if a particular risk was checked. The SoDIS process differs because it uses software engineering procedures for testing to maintain an accurate record of all elements considered. Thus, explicit

statements of risk-assessed situations can be issued. Those potential risks that are identified explicitly are passed on to Stage 4 for further analysis.

STAGE 4 – IDENTIFICATION AND ARTICULATION OF SOLUTION AND GENERATION OF THE SoDIS

At Stage 4 the analysts address the potential risks to stakeholders and to the project. The input to this stage is the concerns developed in Stage 3

A group of analysts, which may consist of software developers and domain and application specialists, formalize their risk concern. This formalization consists of a specification of the concern and an estimation of the severity of the impact of the risk. A particular identified risk may impact elements of the project at one level and may impact extended stakeholders at another. The analysts record the worst-case severity for each particular risk.

SoDIS uses three broad levels of severity; critical, significant, and minor. Instead of making difficult and debatable quantitative judgements, the analyst uses one of these qualitative categories. These categories are used later to prioritise the risks into an order in which they need to be addressed and further analyzed. This procedure meets a management need sometimes addressed by quantified rankings.

If the analysts notice that the risk may also be true for other task-stakeholder relations, they can revisit Stage 3 and apply that concern to the other relationships. In a standard group decision process, the articulation of an issue may bring to mind a heretofore unidentified or misidentified stakeholder. The analyst can return to Stage 1 and add the new stakeholder and then work through Stages 3 and 4 for the new stakeholder.

The relation between Stages 3 and 4 is not a forced linear relation. While in Stage 3 recording a concern, the analyst can also record a possible solution to the problem. If they cannot suggest a solution immediately they can still proceed with the analysis.

The potential solutions can be reordered and tracked similar to the quantitative monitoring and management approach. The process also requires a declaration as to whether the risk mitigation requires new tasks, deleting tasks, or modifying tasks. The resulting Software Development Impact Statement consists of a set of qualitative risks ordered by a degree of severity. Some of these risks can have a probability assigned and others cannot. The project manager can use this information to structure an approach to risk mitigation.

The SoDIS decision making process moves from a coarse granularity to a fine granularity as it iterates through the stages as shown in Table 1. Stage 1 is a coarse high level and often sparse or ill-defined project description which simply sets the scene and informs succeeding stages. Stage 2 provides a detailed or fine description of the project work which informs the succeeding stages. Stage 3 is a detailed low level analysis which identifies potential risks. Finally Stage 4 goes back to the higher level to resolve risks in a systematic manner providing a well-defined account of the risks. At any stage the introduction of new elements – tasks or stakeholders- generates new questions and their impact on the project is ill-defined until an analysis is performed.

Table 1 The SoDIS Process Decision Model

SoDIS STAGE	GRANULARITY	ACTION
1	Coarse – ill-defined	informing
2	Fine	informing
3	Fine – detailed	identifying
4	Coarse – well-defined	resolving

This process of moving from the traditional superficial description of a project to analyse risk in a detailed qualitative manner for each component and then to restate the project based on this new perspective is a new approach. It provides those responsible for the project with a comprehensive perception of associated risks and addresses the limited risk perspective which focuses on high level generic intra-project risks and intra project stakeholders.

The SoDIS process explicitly focuses on risks related to individual tasks and stakeholders. It relies on this analysis process to stimulate the analyst to implicitly recognize risks which may be caused by the interaction of tasks. Research is currently underway, especially in the development of medical software, to incorporate relationship analysis into the SoDIS process. The current SoDIS process relies on the analyst's familiarity with the project to capture interaction risks. The identification of potential risks generated by the interaction of discrete tasks and generated by the interaction of stakeholder groups is topic for further research.

VII. CONCLUSION

Careful application of traditional risk analysis results in limited success in mitigating project failures. In traditional risk analysis the categorization of software projects is limited to internal project characteristics such as project size and complexity. Furthermore it is narrowly focused on internal project stakeholders and emphasized quantifiable risk factors to the exclusion of qualitative ones. Thus traditional risk analysis is either inadequate or incomplete. The SoDIS process complements and completes traditional risk analysis by specifically addressing these problems of traditional risk analysis. The resulting Software Development Impact Statement (SoDIS) provides a snapshot of the risk potential of the planned tasks before undertaking system implementation This pre-audit provides an opportunity to address the risks by mitigation or avoidance. Its use provides checkpoints in the project that enable the software developer to stop or modify a project before disaster strikes. The use of qualitative best practice questions associates a full range of stakeholders with the project tasks providing a comprehensive risk analysis which helps identify social, professional and ethical risks for a project. SoDIS is the first fully-developed approach of this kind. It points the way to achieving successful software development by design rather than by accident.

ACKNOWLEDGEMENT

The Software Development Research Foundation (sdreasearch.org) promotes the use of the SoDIS process and makes a SoDIS software tool available without cost. Research on this process and software development was partially funded by National Science Foundation Grant NSF 9874684 and the work in New Zealand was supported by grants from NACCQ, AUT Faculty of Business, AUT School of Computer and Information Sciences, and Kansas State University. East Tennessee State University also supported Professor Gotterbarn's work in New Zealand. SoDIS research for the Deputy Prime Minister "The Implementation of Electronic Voting" was funded by the UK Government (LGR 65/12/72). The Internet filter example was called to our attention by Irene Davey.

Editor's Note: This article was received on September 13, 2004 and was with the authors for four months for one revision. The article was published on June 18, 2005.

REFERENCES

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their computer or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.

2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.

3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.

4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

Ackoff, R. L. (1974). *Redesigning the Future*. New York: Wiley.

Aladwani, A. M. (2002) "IT Project Uncertainty, Planning and Success: An Empirical Investigation From Kuwait," *Information Technology*, (15)3, pp. 210-226.

Boehm, B. (1989*) Risk Managemen*t, Los Alamitos, CA: IEEE Computer Society Press

Boehm, B. et al, (1998) "Using the WINWIN Spiral Model: A Case Study," *Computer* (31) 7 33-44.

Cooper R. *Winning at New Products: Accelerating the Process from Ideas to Launch,* 3rd edition, Cambridge, MA: Perseus Publishing

Checkland, P. and S. Howell (1998) *Information, Systems and Information Systems,* Brisbane: Wiley.

Cole, A.(1995) "Runaway Projects: Cause and Effects," *Software World (UK), (26)3,*

Dalcher, D. and C. Tully, (2002) "Learning from Failures," *Software Process Improvement and Practice*, (7) 2 pp 71–89.

Demarco, T. and T. Lister, *In Waltzing with Bears: Managing Risks of Software Projects*, NY:Dorsett House 2003

Dey, P. K. (2002) "Benchmarking Project Management Practices of Caribbean Organizations Using Analytic Hierarchy Process," *Benchmarking: An International Journal*, (9) 4, pp 326-356

Farbey, B, F. Land, and D, Targett, (1993) *How to Assess Your IT Investment*, Oxford: Butterworth Heinemann.

Fidler, C. and S. Rogerson, (1996) *Strategic Management Support Systems*. London: Pitman Publishing.

Goodpaster, K..E. (1993) "Business Ethics and Stakeholder Analysis," in Beauchamp, T.L. and N.E. Bowie (eds) *Ethical Theory and Business*,. Upper Saddle River, NJ: Prentice Hall, 85-93.

Gotterbarn, D. (2004) "Reducing Software Failures Using Software Development Impact Statements" in R. Spinello and H. Tavani (eds.) *Readings in Cyber Ethics* 2nd edition. Sudbury, Mass: Jones and Bartlett Publishers, Inc

Gotterbarn D., T. Clear, and C. Kwan, (2004) "An Inspection Process for Managing Requirements Risks: Software Development Impact Statements" *Proceeding of NACCQ Conference,* Christchurch NZ

Hoffman G. (2003),"Integrating PSP and CMMI Level 5", http://www.stc-online.org/stc2003proceedings/PDFFiles/pres1001.pdf (current 2/2/2005)

Henry, J. (2004) *Software Project Management, A Real-World Guide to Success,* Boston, MA: Addison Wesley.

Highsmith, J. (2004) *Agile Project Management,* Boston, MA: Addison Wesley

Hirsheim R. and H. Klein (1989) "Four Paradigms of Information Systems Development" *Communications of the ACM*, (32)10, pp 1199-1216.

Humphrey, W. (1989) *Managing the Software Process*, Boston, MA: Addison-Wesley

Humphrey, W. (1996) *Introduction to the Personal Software Process,* Boston, MA: Addison-Wesley.

Humphrey, W (1999) *Introduction to the Team Software Process,* Boston, MA: Addison-Wesley

Jayaratna, N. (1994) *Understanding and Evaluating Methodologies*. Maidenhead, UK : McGraw-Hill,.

Jones, C. (2000) *Software Assessments, Benchmarks, and Best Practices*, Boston, MA: Addison-Wesley

Keil, M., T. Amrit, and A. Bush, (2002) "Reconciling User and Project Manager Perceptions of IT project risk: A Delphi Study," *Information Systems Journal.* (12) 2, pp 103-119.

Keil M. et al, (1998) "A Framework for Identifying Software Project Risks", *Communications of the ACM* (41) 11.

Keil M. et al,. (2000) "An Investigation of Risk Perception and Risk Propensity on the Decision to Continue a Software Development Project," *Journal of Systems and Software,* (53) 2, pp145-157

Kuruppuarachchi, P. R., P. Mandal and R. Smith (2002) "IT Project Implementation Strategies for Effective Changes: A Critical Review," *Logistics Information Management*, (15) 2, pp126-137.

Laudon, K. C. and Laudon, J. P. (2005) *Essentials of Management Information Systems: Managing the Digital Firm,* sixth edition, Upper Saddle River, NJ: Pearson Prentice Hall, pp428-430.

Lyytinen K. and R. Hirschheim (1987) "Information Systems Failures-A Survey and Classification of Empirical Literature," *Oxford surveys in Information Technology,* (4) pp 257-309

Mackenzie, K., (2001) "IT A Necessary Evil: Survey," *The Australian*, 21 August.

Mitchell K.R., R.B. Agle and J.D.Wood, J.D (1997) "Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who or What Really Counts," *Academy of Management Review* October (22) 4, p 853.

Mumford, E. (1996) *Systems Design: Ethical Tools for Ethical Change*, Basingstoke, UK: Macmillan Ltd.

Papazafeiropoulou A., A. Pouloudi, and A. Poulymenakou, (1995) "Use of Stakeholder Analysis for Electronic Commerce Applications in the Public Sector: Different Development Scenarios," in J.Pries-Heje et al., editor, *7th European Conference on Information Systems (ECIS99)*, pages 895-908, 1999. Copenhagen.

PMI (2000): Project Management Institute – *Member Ethical Standards and Member Code of Ethics,* http://www.pmi.org/membership/standards (current May 2005)

Pouloudi A., and E.A. Whitley (1997) "Stakeholder Identification in Inter-Organizational Systems: Gaining Insights for Drug Use Management Systems," *European Journal of Information Systems* (6) 1, pp. 1-14.

Prestwood Software (2002) "Prestwood Software Development Process, Overview," Version 3.0 R1 July 2002 www.prestwood.com/standards/psdp/downloads/PSDP%20Overview.pdf (current May 2005).

Ropponen J., N. Lyytinen and N. Kalle,(2000) " Components of Software Development Risk: How to Address them? A Project Manager Survey.," *IEEE Transactions on Software Engineering* (26)2, pp. 98-112.

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

Schmidt R. et al., "Identifying Software Project Risks: An International Delphi Study," *Journal of Management Information Systems* (17)4, pp 5-35.

Schwalbe, C. (2004) *Information Technology Project Management,* 3rd ed. Boston, MA: Thomson Publishing.

SEI (Software Engineering Institute, Carnegie Mellon University) (1995) *The Capability Maturity Model: Guidelines for Improving the Software Process*, Boston MA: Addison Wesley Professional,.

Sharp, H. et al. (1999) "Stakeholder |Identification in the Requirements Engineering Process," *DEXA Workshop Proceedings*, 1999, pp 387-391

Shenhar, A.L. J.J. Renier, and R.M. Wideman, (1996) "Improving PM: Linking Success Criteria to Project Type." In *Creating Canadian Advantage through Project Management*, Project Management Institute Symposium. Calgary, May.

Shneiderman B and A. Rose (1995) "Social Impact Statements: Engaging Public Participation in Information Technology Design," *Technical Report of the Human Computer Interaction Laboratory*, September, pp 1-13.

Simon, H.A. (1960) *the New Science of Management Decision*. Engelwood Cliffs, NJ: Prentice Hall.

Simon, H.A. (1984) "Decision Making and Organisational Design." In D.S. Pugh, (ed) *Organization Theory*. Penguin Books Ltd. pp 202-223.

Smith, H.J. and M. Keil, (2003) "The Reluctance to Report Bad News on Troubled Software Projects: A Theoretical Model," *Information Systems Journal* (13) 1, pp 69-95.

Spafford, G.(2002) "Staking Out the Stakeholders," *Gantthead.com* October 2002 (current 15 May 2003)

Turban, E., E. Mclean, and J. Wetherbe, (1996) *Information Technology for Management*, Improving Quality and Productivity . New York: John Wiley.

Willcocks, L., & D. Mason, (1987). *Computerising Work:People, Systems Design and Workplace Relations*. London:Paradigm.

ABOUT THE AUTHORS

Don Gotterbarn is the Director of the Software Engineering Ethics Research Institute at East Tennessee State University and teaches computer ethics, software engineering, and software project management. He is a visiting professor at the Centre for Computing and Social Responsibility in England. He worked as a computer consultant on software projects for the U.S. Navy and for the Saudi Arabian Navy. He has also worked on the certification of software for vote counting machines and missile defense systems. His technical work includes funded research on performance prediction for a distributed Ada closure, object-oriented testing, and software engineering education and computer ethics. He was awarded the "Making a Difference" award by the ACM special interest group on Computing and Society for his work in promoting professionalism in the teaching and practice of software development.

Simon Rogerson is Director of the Centre for Computing and Social Responsibility at De Montfort University, UK and Europe's first Professor in Computer Ethics. Following a successful industrial career in information systems development, he now combines research, lecturing and consultancy in the management, organisational and ethical aspects of information and communication technologies. Simon advised the European Commission on ICT social policy and the Russian Government on the implications of the information society. In the UK he was a leading member on the Measures of Success project for the e-Envoy and the Implementation of Electronic Voting project as well as advising the government on the use of ICT to address social inclusion. Simon was the winner of the 1999 IFIP Namur Award for outstanding contribution to the creation of awareness of the social implications of information technology. In 2003 he was a finalist for the World Technology Award in ethics.

Communications of the Association for Information Systems

ISSN: 1529-3181