Keynote: Understanding and Reducing Project Failure: The Ethics of

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

ABSTRACT

The common approach to risk analysis and understanding the scope of a software project has contributed to significant software failures. A process is presented which expands the concept of software risk to include social, professional, and ethical risks that lead to software failure. Using an expanded risk analysis will enlarge the project scope considered by software developers. A tool to develop Software Development Impact Statements is also discussed.

KEYWORDS:

Project Management, Risks, SoDIS, Ethical issues

1. INTRODUCTION

Software engineering has been evolving and refining techniques to help produce software products that meet the needs of their clients. These techniques emphasize a specific set of risks -missed schedule, over budget, and failing to meet the system's specified requirements. Nevertheless, software development has been characterized as a "software crisis". A high percentage of software is being delivered late, over budget, and not meeting all requirements.

In this paper I correct the meaning of "Software Failure", or more precisely, focus attention on some overlooked meaning of "Software Failure". Software fails even though it was produced on schedule within budget and met the customer's specified software requirements. Software has been developed which, although meeting stated requirements, has significant negative social and ethical impacts. By ethical impact I mean those impacts of software which positively or negatively the circumstances, experiences, behavior, livelihood, or daily routine of others. The ethical stakeholders in software are those who are so affected.

The Aegis radar system, for example, met all requirements that the developer and the customer had set for it. The system designer's did not take into account the users of the software nor the conditions in

which it would be used. The 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 Vincennes shooting down an Iranian commercial airliner killing 263 innocent people.

There are two factors that contribute to these professional and ethical failures. There is significant evidence that many of these failures are caused by limiting the consideration of relevant system stakeholders to just the software developer and the customer. This leads to developing systems that have surprising negative affects because the needs of relevant system stakeholders were not considered. In the case of the Aegis radar system the messages were not clear to the users of the system operating in a hostile environment. These types of failures also arise from the developer limiting the scope of software risk analysis just to technical and cost issues. A complete software development process requires the identification of all relevant stakeholders and broadening the risk analysis to address social, political, and ethical issues. Software development processes include a risk analysis process but with current methods limit the types of risks considered. The risk analysis is primarily instrumental-addressing corporate bottom lines. Software projects have ethical dimensions that need to be identified before and during the development process. There are some modifications to the standard development models that will address these additional types of risk.

There are some techniques that attempt to include a broader consideration of stakeholders, such as viewpoint requirements definition. Some of these software development methods articulate a distinction between direct system stakeholders—(those who)"receive services from the system and send control information to the system"-and indirect stakeholders— those who "have an interest in some of the services that are delivered by the system but do not interact directly with it". These would include the passengers on the Iranian airline or the driver of an automobile whose breaks are controlled by a computer program. Unfortunately 1) these methods do not provide a way to reach beyond identifying those who have a business relation to the customer. They would not have identified as indirect stakeholders the 47 people killed by falling debris from a Patriot missile. These methods also fail to 2) provide a method of identifying the social and ethical impacts on the indirect stakeholders.

Barry Boehm's has developed a methodology which comes close to meeting the first factor, the stakeholder identification problem. His Win-Win spiral software development technique is used to elicit project requirements for all stakeholders. At each phase of a project's development the analyst identifies the stakeholders for that stage, determines the win conditions for each new stakeholder, and then negotiates to have these new win condition requirements fit into a set of Win-Win conditions that have already been established for all concerned. There is a set of win conditions for the Aegis radar customer. These conditions would be identified and a process developed to meet those conditions. Then new stakeholders would be identified, for example the sailor's using the system on the Vincennes, and their win conditions would be identified. They would consider it important to be able to clearly determine if an approaching aircraft were hostile. This win-condition would be incorporated, via negotiation, into the existing process plan. There is no methodology to identify ethically relevant stakeholders nor is there an ethical foundation for the negotiation process.

The method is also limited in that it assumes all stakeholders are equal and that they will equally be aware of and able to describe their own win conditions. The negotiation amongst stakeholders will be unjust and will likely lead to a failed systems, unless, contrary to fact, each stakeholder has such an equal identification and descriptive skill of their own win conditions. There is also an implicit assumption that all requirements are negotiable. As the method is constructed, all requirements have equal status-none are rejected because they are morally impermissible or required because they are morally mandatory.

The major portion of the paper develops a methodology to help software engineers address the ethical issues that lead to failed systems. The methodology contains a technique for stakeholder identification and an approach to ethical analysis in software development that avoids many difficulties with business ethics methods of stakeholder identification that fail to capture requirements that emerge from the relationship between stakeholders. The goal of the method is to help the software engineer identify

all of the ethically relevant stakeholders and provide structure to the process through a series of ethics principles.

Software, which has been developed to test the feasibility of this method, will also be presented.

2. CURRENT BEST? PRACTICE

Current software project management techniques are used to develop and deliver various types of software products. In developing a software project management plan decisions are made about technical issues such as: a) which software development methodology to use, b) which cost and estimation techniques to use, c) how to reduce risk, and e) which programming development environment to use. The software project management plan is used to control all aspects of the software development process.

The literature is rife with stories about systems that failed, some which merely inconvenience people or cost money, while others are much more significant. There are some wonderful examples of software failures. Recall the infamous example of the Australian Department of Defense's reuse of code from an infantry simulation program to model the movement of kangaroos disturbed by approaching helicopters. The simulation worked to show the kangaroos running for cover as the helicopter approached, but it also continued with the infantry model and showed the kangaroos regrouping and coming back over the hill armed with bazookas and missile launchers attacking the disruptive helicopter. This was simulation but code does get used in real applications which are not so humorous. A New Jersey inmate under computer-monitored house arrest removed his electronic anklet. "A computer detected the tampering. However, when it called a second computer to report the incident, the first computer received a busy signal and never called back." [Joch] While free, the escapee committed murder. In another case innocent victims were shot to death by the French police acting on an erroneous computer report[Vallee].

2.1 Bad Science

We do not try to produce failed software. What is the problem? We are told that part of the problem is the nature of software development. We are forced to set milestones before we fully understand the requirements. So we develop techniques to focus on the requirements, the technical documentation for functional requirements of a system. We can even describe these requirements with mathematical rigour. But we do not address anything beyond their technical feasibility. We have developed an alphabet soup approach to address these problems. We have the CMM, the PSP and the TSP. These are directed at broad long term needs of software developers and software development and directed at improving productivity and efficiency. Soon we will have ESP. The problem is that we apply these methods with the best intentions and we still have failed projects. Bad science is described as repeating the same experiment over and over and expecting different results. We are appalled when students simply recompile code over and over without making any changes in the hope that doing it one more time will change something. Bad software development is applying the same methods of software development and risk analysis over and over and expecting different results.

2.3 Failure Research

What does failure research say is wrong? There are numerous infamous cases of software failure. There are multiple causes of these failures, but they did have at least on common elements. 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 regarding evaluation of IT investment, "... 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. We should not limit our consideration of stakeholders to those who are financing the project or politically influential. Stakeholders are individuals or groups who may be directly or indirectly affected by the project and thus have a stake in the development activities. Those stakeholders who are negatively affected are particularly important.

Negative effects include both overt harm and the denial or reduction of goods. So obviously the development of medical software that delivers erroneous dosages of medicine that killed patients would have a negative effect; but we would also include as having a negative effect software that limited people's freedom of expression. Limitations on positive ethical values and rights are negative effects.

Many companies have gone out of business because they have only emphasized short term efficiency and productivity. The quantity and cost of major product recalls in terms of dollars and company reputation are evidence of this mistaken emphasis on short term goals. 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 the technical elements of a piece of software. This self-imposed limitation has contributed to the development of software that has been inferior and has had negative consequences for others: software that is not socially sensitive. The systems we develop perform tasks that affect other people in significant ways. The production of quality software that meets the needs of our clients and 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.

Frequently the failure to consider social, ethical, and other risks has led to the delivery of unacceptable software that should be recalled and modified. Because the process of recall and modification is too expensive for the developer, the product remains on the market. The scope of a project needs to be identified in terms of its real stakeholders.

The expansion of the scope of a project to include all relevant stakeholders will also broaden the types of risks considered. Many companies have gone out-of-business because they have only emphasized short-term efficiency and productivity. The quantity and cost of major product recalls in terms of dollars and company reputation is evidence of this mistaken emphasis on short-term goals. 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. The systems we develop perform tasks that affect other people in significant ways. The production of quality software that meets the needs of our clients and 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.

We need to extend the traditional software project stakeholder list from customers and corporations or shareholders to include all those who will be affected by the software and by its production. This enlargement of the domain of stakeholders has been implicitly endorsed by professional societies in the paramouncy clause — " Protect public health, safety, and welfare" in their codes of ethics. This extension has been explicitly adopted in several legal decisions in the United States. This extended domain of stakeholders includes: users of the software, families of the users, social institutions which may be radically altered by the introduction of the software, the natural environment, social communities, software professionals, employees of the development organization and the development organization itself. Given such a range of stakeholders, how is one ever to learn how to identify the relevant and significant stakeholders?

3. RISK ANALYSIS WITH SOFTWARE DEVELOPMENT IMPACT STATEMENTS

Funded research has been done on the development of a risk management process employing software development impact statements. 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. A SoDIS is intended to reflect both software development process and the more general obligations to various stakeholders. Although all software projects have some unique elements, there are significant similarities between projects so that a generic practical approach can be taken to refocus the goal of a project to include a consideration of all ethically relevant stakeholders as well as all technically relevant stakeholders.

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.

We can divide software project development into three distinct phases. They are: the Feasibility phase that includes considerations of preparedness to start a project and managing action items needed to start the project; the Requirements Phase that defines the specifications of a system and identifies

Figure 1:

The Analysis Phases

and manages potential risks with each requirement; and the Detailed phase that uses a detailed software project management plan to manage each task on system development. Each of these phases has its own peculiar risks. The purpose of the SoDIS Project Auditor is to identify these risk in a pre-audit of each phase.

In the Requirements phase, we can develop a high level analysis of the expected impacts of a project. A detailed SoDIS is developed from a preliminary Gantt chart. The goal of the SoDIS process is to identify significant ways in which the completion of individual tasks may negatively affect stakeholders and to identify additional project tasks needed to prevent any anticipated problems.

4. THE SoDIS PROCESS

On a high level, the SoDIS process can be reduced to four basic steps: (1) the identification of the immediate and extended stakeholders in a project, (2) the analysis of the tasks or work breakdown packages in a project, (3) for every task, the identification and recording of potential ethical issues violated by the completion of that task for each stakeholder, and (4) the recording of the details and solutions of significant ethical issues which may be related to individual tasks and an examination of whether the current task needs to be modified or a new task created in order

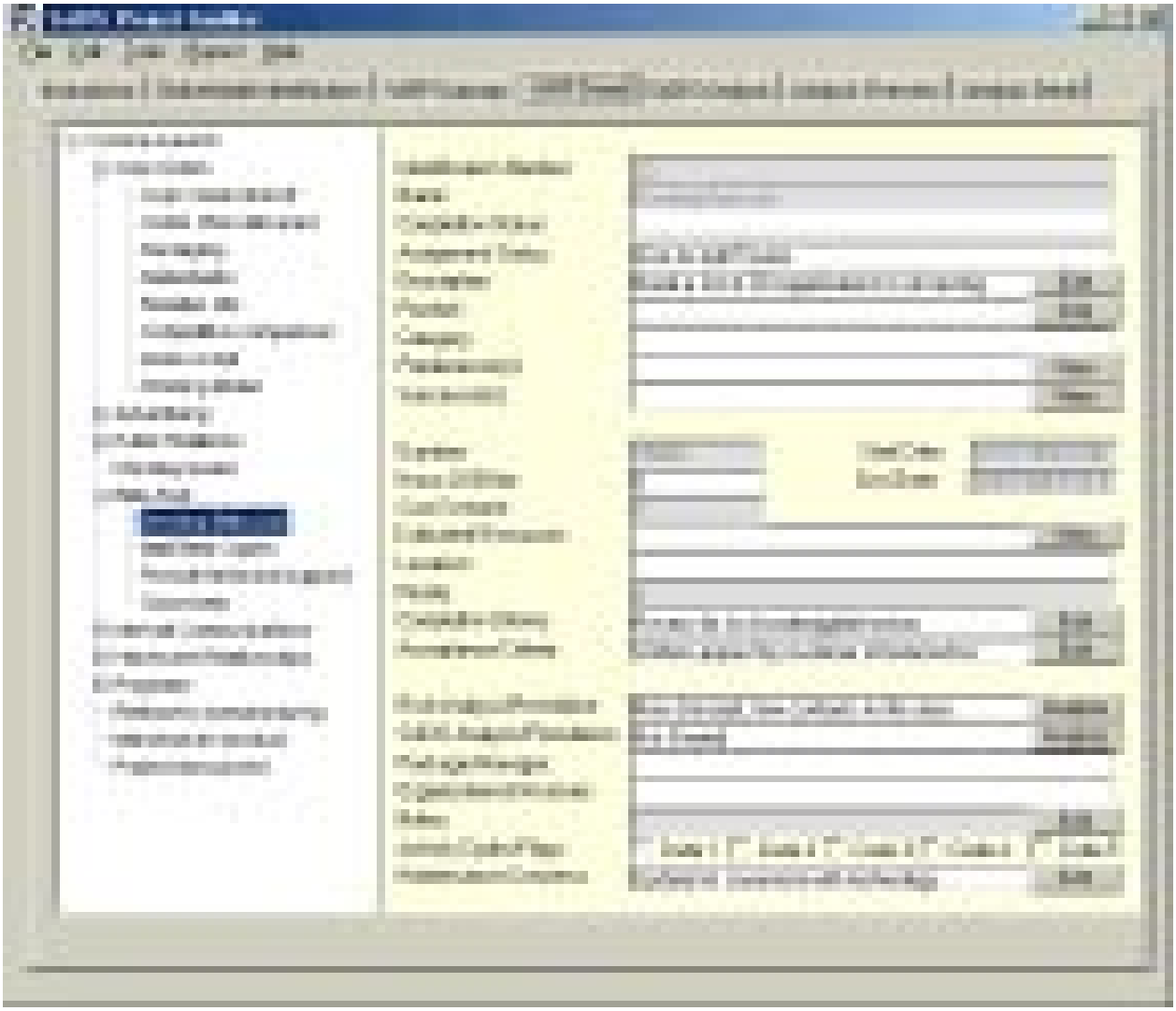

Figure 2:
WBP Detail Screen

to address the identified concern.

To aid with the major clerical task of completing this process for every task and for every stakeholder a tool - the SoDIS Project Auditor - was developed. The SoDIS Project Auditor keeps track of all decision make about the impact of project tasks on the relevant project stakeholders and it enables a proactive way to address the problems identified.

4.1 Work Breakdown Structure

Most software project management models proceed by decomposing the project into component tasks called "work breakdown packages" that 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.

Each of these individual tasks may have significant ethical impact. The specific SoDIS is used to help the developer responsibly address the ethically loaded potential of each work breakdown package. This is accomplished by including a SoDIS analysis in the standard descriptive elements of a work breakdown package (figure 2).

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 software project management plan. The early identification of these software modifications saves the developer time and money and leads to a more coherent and ethically sensitive software product. This phase of the SoDIS process is a pre-audit of a detailed project plan that is developed late in a software development life cycle.

4.2 Stakeholder Identification

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 behavior, 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.

Stakeholders are also those to whom the developer owes an obligation. The imperatives of the Software Engineering Code of Ethics and Professional Practice and similar codes define the rights of the developer and other stakeholders. These imperatives can be used to guide the stakeholder search. The process of identifying stakeholders also identifies their rights and the developers' obligations to the stakeholders. Many of the computing codes have similar imperatives. These have been reduced and categorized under five general principles in the SoDIS process and incorporated into the SoDIS Project Auditor.

The SoDIS process also includes a consideration of other phases of an SDLC. Some risks can be identified when a project is first conceived or can be identified at an intermediate stage when the customer's desires are being specified in the requirements phase. The SoDIS Project Auditor also provides a pre-audit for these two project phases.

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 a SDLC.

The identification of stakeholders must strike a balance between a list of stakeholders that includes people or communities that are ethically remote from the project, and a list of stakeholders that only includes a small portion of the ethically relevant stakeholders. Rogerson & Gotterbarn had proposed a method to help based on Gert's moral rules [Gert 1988]. Gert gives 10 basic moral rules. [Gotterbarn 1991] These rules include: Don't kill, Don't cause pain, Don't disable, Don't deprive of freedom, Don't deprive of pleasure, Don't deceive, Don't cheat, Keep your promises, Obey the law, and Do your duty. These rules carry with them a corresponding set of rights such as the right to liberty, physical security,

	Customer	Developer	User	Community	Additional stakeholders.....
Req\Stakeholder					
Requirement 1	N	N	N	N	
Requirement 2	N	N	N	Y	
Requirement 3	Y	N	Y	Y	

Might the completion of this requirement cause harm to the stakeholder? ('Y' indicates that the task may cause harm to the stakeholder group)

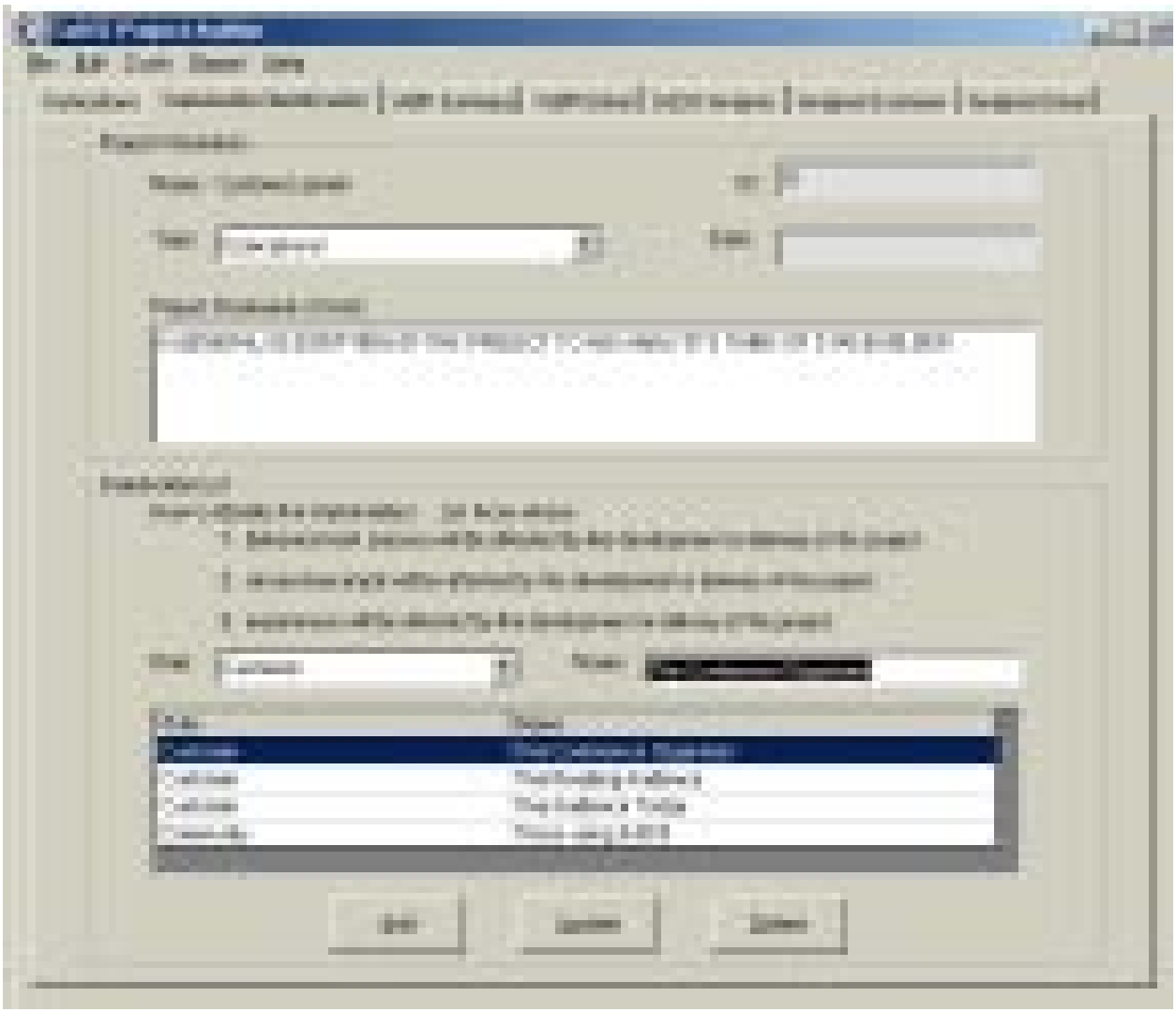

Figure 3:

Stakeholder Identification

personal liberty, free speech, and property. How can these rules be used to identify stakeholders?

A matrix can be set up for each ethical rule such as “Don’t cause harm.” The column headers of the “Don’t cause harm matrix” are the stakeholders, such as the “developer” and the “customer”, and there is a row for each major requirement. The SoDIS analysts then visits each cell in the matrix asking, for each requirement whether meeting this requirement violates that obligation to the stakeholder. Because the analysis as described is organized by particular software requirements, it will be easy to identify those requirements which generate a high level of ethical concern. Thus, the list will also be used to determine if particular requirements have to be modified to avoid significant ethical problems. This method can be used at this stage to give a composite picture of the ethical impact of the entire project from the point of view of these stakeholders.

This process is now used to both identify additional stakeholders and to determine their rights The first phase of the stakeholder identification should have identified some areas of broader ethical concern and some additional stakeholders. The primary stakeholder analysis is repeated for these newly identified stakeholders. Even if there were no new stakeholders identified, at a minimum the analysis should include software users, related cultural

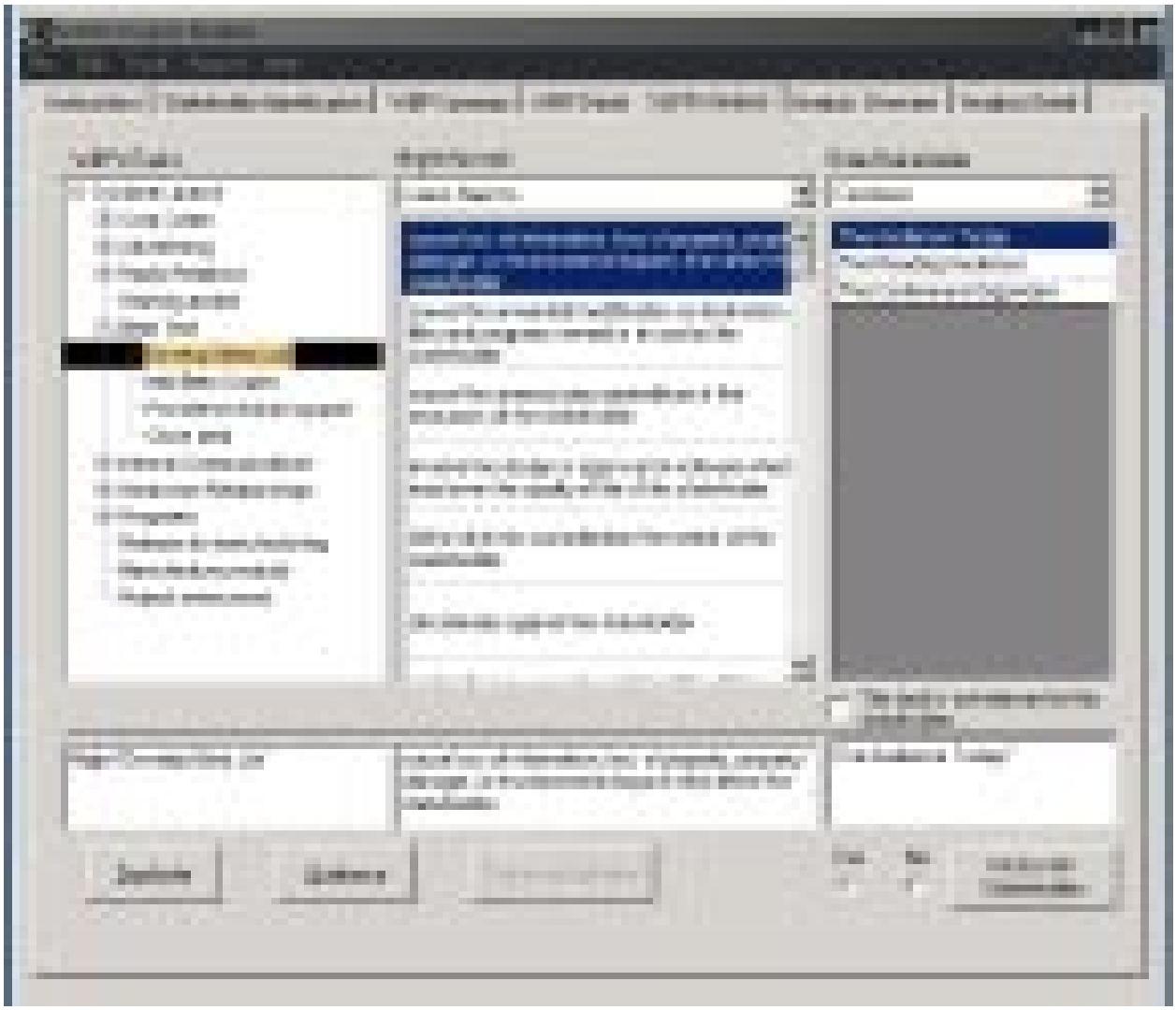

Figure 4:

SoDIS Analysis screen

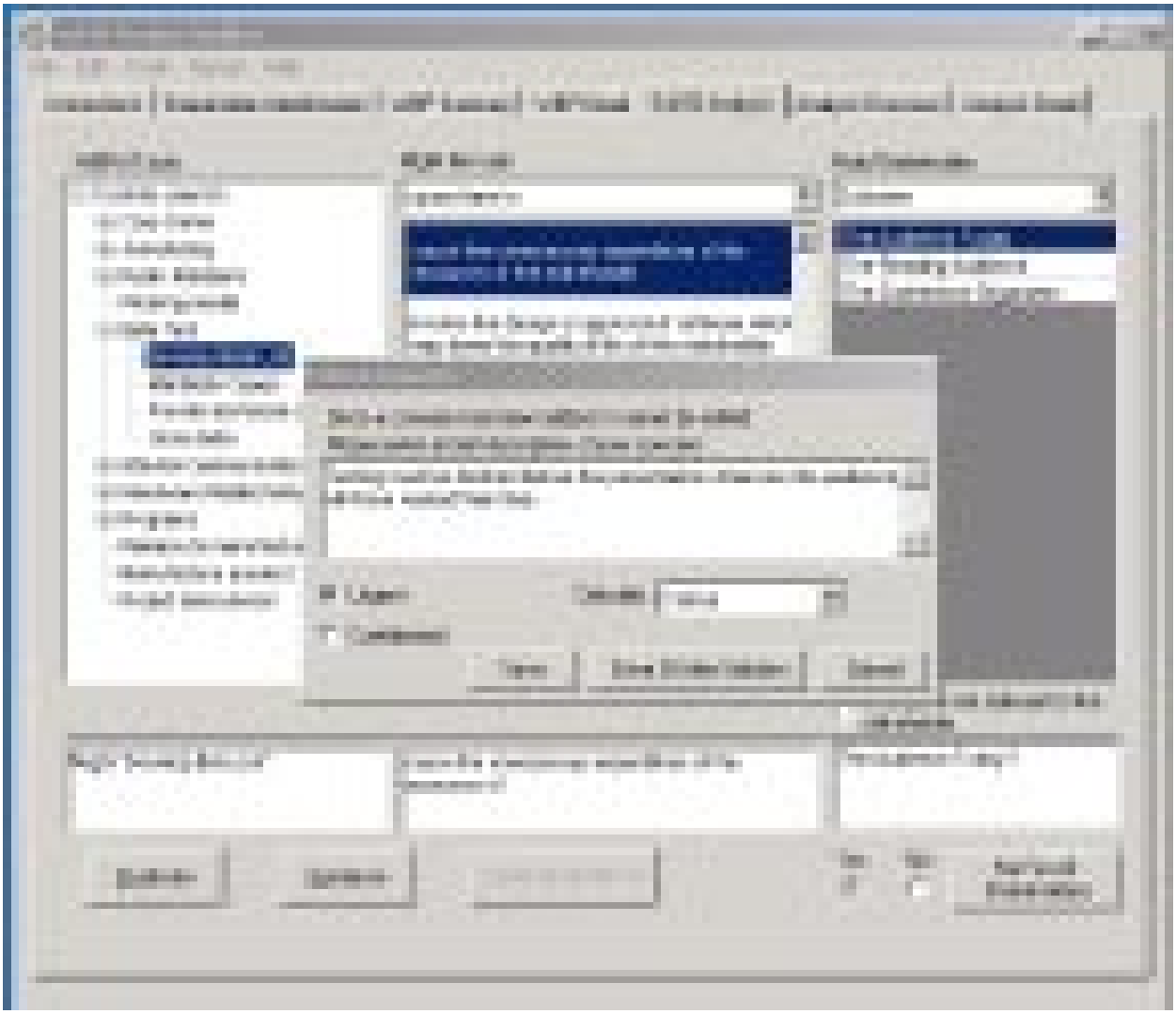

Figure 5:
Concern /Solution Screen

groups, and society as potential stakeholders.

The system provides a standard list of stakeholders that are related to most projects. This standard list of stakeholder roles changes 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 stakeholders roles.

The stakeholder identification form (figure 2) contains a Statement of Work that helps remind the analyst of the project goals and facilitates the identification of relevant stakeholders. 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 which will now include the new stakeholder.

4.3 Ethical Obligations

This stakeholder identification process has been modified in the SoDIS Project Auditor. Gert's ethical principles have been combined with ethical imperatives from several computing codes of ethics to reflect the professional positive responsibility

of software developers. These principles have been framed as a set of 32 questions related to stakeholders in a software project, and to generalized responsibility as a software professional.

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. When the analysis is complete there are several usage statistics reports that give various snapshots of the major ethical issues with the project.

When an ethical concern has been identified, the analyst gets an ethical concern form which 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 analyzed. If the problem is significant then they have to determine whether the problem requires a modification of the task, deletion of the task from the project, or the addition of a task to overcome the anticipated problem. It is these adjustments to the software requirements or management project plan that complete risk analysis.

The process of developing a SoDIS requires the consideration of ethical development and the ethical impacts of a product — the ethical dimensions of software development. The SoDIS analysis process also facilitates the identification of new requirements or work breakdown packages that can be used as a means to address the ethical issues. The identified work breakdown packages need to be incorporated into the software project management plan. The early identification of these software modifications saves the developer time and money, and leads to a more coherent and ethically sensitive software product.

CONCLUSION

The SoDIS process facilitates the expansion of software risk analysis to reduce software failures. Using this pre-audit process in test in the UK and the USA facilitated the early identification of project risks. Using a SoDIS process will make producing software of high quality and producing software that is ethically sensitive second nature to the software engineer.

This research was partially funded by NSF Grant 9874684

REFERENCES

Collins W R, Miller K W, Spielman B J and Wherry P (1994) How Good is Good Enough, Communications of the ACM, Vol 37 No 1, January, pp 81-91.

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

Gert B (1988) Morality, Oxford University Press.

Gotterbarn D (1991) Computer Ethics: Responsibility Regained, National Forum, The Phi Kappa Phi Journal, Vol 71 No 3.

Gotterbarn D (1999) "Promoting Ethical responsibility in Software Development," Proceeding of the AICE Computer Ethics Conference

Gotterbarn D and Miller K and Rogerson S (1999) Software Engineering Code of Ethics, Communications of the ACM. 1998
http://computer.org/computer/code-of-ethics.pdf

Green R M (1994) The Ethical Manager, Macmillan Publishing.

Joch A (1995) "How Software Doesn't Work," Byte, December 1995 pp 48-60.

McCarthy J (1996) Dynamics of Software Development, Microsoft Press.

O'Connell F (1994) How to run successful projects, Prentice-Hall.

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

Vallee J (1982) The Network Revolution, Berkeley: And/Or Press 1982).