REDUCING SOFTWARE FAILURES: ADDRESSING THE ETHICAL RISKS OF THE SOFTWARE DEVELOPMENT LIFECYCLE

By Don Gotterbarn
Software Engineering Ethics Research Institute,
East Tennessee State University, USA

ABSTRACT

A narrow 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. This process also is incorporated into a software development life cycle. A tool to develop Software Development Impact Statements is also discussed.

INTRODUCTION

Software developers and software engineers have been evolving and refining techniques to mediate risks of developing software products that meet the needs of their clients. The risks focused on include: missed schedule, over budget, and failing to meet the system's specified requirements. In spite of this attention to risks, a high percentage of software is being delivered late, over budget, and not meeting all requirements, leading to software development being characterized as a "software crisis".

SOFTWARE RISK

The Problem

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 when it is produced on schedule within budget and meets 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 people.
There are two factors that contribute to these professional and ethical failures; both related to those who have something to gain or lose as a result of the software project — system stakeholders. First, there is significant evidence that many of these failures are caused by limiting the consideration of system Stakeholders to just the software developer and the customer. This limited scope of consideration 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. Second, these types of failures also arise from the developers limit the scope of software risk analysis just to technical and cost issues. A complete software development process requires 1) the identification of all relevant stakeholders and 2) enlarging risk analysis to include social, political, and ethical issues. I propose to add a process to a standard life cycle model that will help identify the relevant stakeholders and broaden the scope of risks anticipated.

Software Quality and its Risks

The primary goal of software developers is the production of quality systems that meet the needs of the user. "Software quality" is defined in terms of customer satisfaction. "Risk" is understood as any potential threat to the delivery of a quality product. To meet the goal of quality software, developers focus on particular risks including: project and schedule slips, cost increases, technical and quality risks, the timeliness of the product, risks that the final product will not fit the business for which it was designed.
Projects are managed focusing on these risks. Tools used to help identify and manage these risks include: risk tables, and lists of risks categorized by type, probability and impact. The checklist process is reminiscent of the process pilots go through before take off. As airline passenger we are made more comfortable by the fact that they go through this procedure. But unlike pilots, developers choose to ignore some risks. Risks levels are

determined based on the anticipated impact of the risk and its probability of occurring. Only risks above limited specified levels are addressed.

Software Development Life Cycles (SDLC) and Risk

There are several models for developing software that reduce planning risks. All of these models contain similar phases: develop a statement of the customer's desires – the requirements phase; design how to achieve those desires – the design phase; code and test what was designed – the implementation phase; and determine that the requirements are satisfied by the system developed – system testing phase. The ordering and content of the steps through these phases called the system development life cycles (SDLC). Many SDLCs are linear and require the documented completion of a one phase before going on to the next phase and are directed at the satisfaction of the customer's explicit requirements. Only a few SDLCs include any risk analysis or expand the number and type of system stakeholders considered.

Spiral Lifecycle and Risk

Barry Boehm's "spiral lifecycle" is one of the few SDLCs that specifically address risk. Following his model, software is developed in a series of incremental releases. Each iteration through the spiral includes tasks related to: Customer communication, Planning, Risk Analysis, Engineering the development of the next level, Construction and Release and, Customer evaluation and assent. Each incremental element of the product that passes through these phases has undergone risk analysis and evaluation by the customer. Although this model introduces a focus on risks, those risks are limited to the risks identified above.

Lifecycle Measurement (LIME) and Stakeholders

LIME [Buglione, 1999] introduces a multidimensional analysis of quality and performance during software development. This model defines quality in terms of an economic dimension from the managers' viewpoint with particular attention to cost and schedule drivers; a social dimension from the users viewpoint, and a technical dimension from the developer's viewpoint with particular attention to technical quality that has a different impact during each SDLC phase. This model examines the role of a stakeholder in all phases of the project development. Although this method includes a consideration of a stakeholder in all phases of project development, the stakeholder is the user for whom quality is achieved by the satisfaction of the specified requirements.

THE REAL PROBLEM

Limitations of LIME and the Spiral SDLC

Both of these models are improvements over earlier SDLCs that either had no risk analysis or had a very limited view of system stakeholders. The Spiral model incorporates risk analysis and LIME incorporates a consideration of an additional stakeholder throughout the development process. The risks considered by the Spiral model are the technical risks identified earlier and LIME expands the risk analysis to include risks to one stakeholder in terms of requirements specification.
All of these methods attempt to anticipate and avoid all potential threats to a software project. The negative possibilities are those that would delay the delivery of the software that performs the desired functions in a timely and cost effective manner. However, none of these methods consider the ethical issues that need to be identified and addressed during the planning stages and re-considered throughout the development process.

Failure Research

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 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 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.

Stakeholder Identification

Some of these software development methods distinguish 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 airliner or the driver of an automobile whose brakes are controlled by a computer program. Unfortunately 1) these methods do not provide an ethical or philosophical foundation for this distinction 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.
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?

SOFTWARE DEVELOPMENT IMPACT STATEMENT

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.
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 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 is to identify these risks 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 software development plan. 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.
A detailed SoDIS is developed from a preliminary Gantt chart. 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. 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. . 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

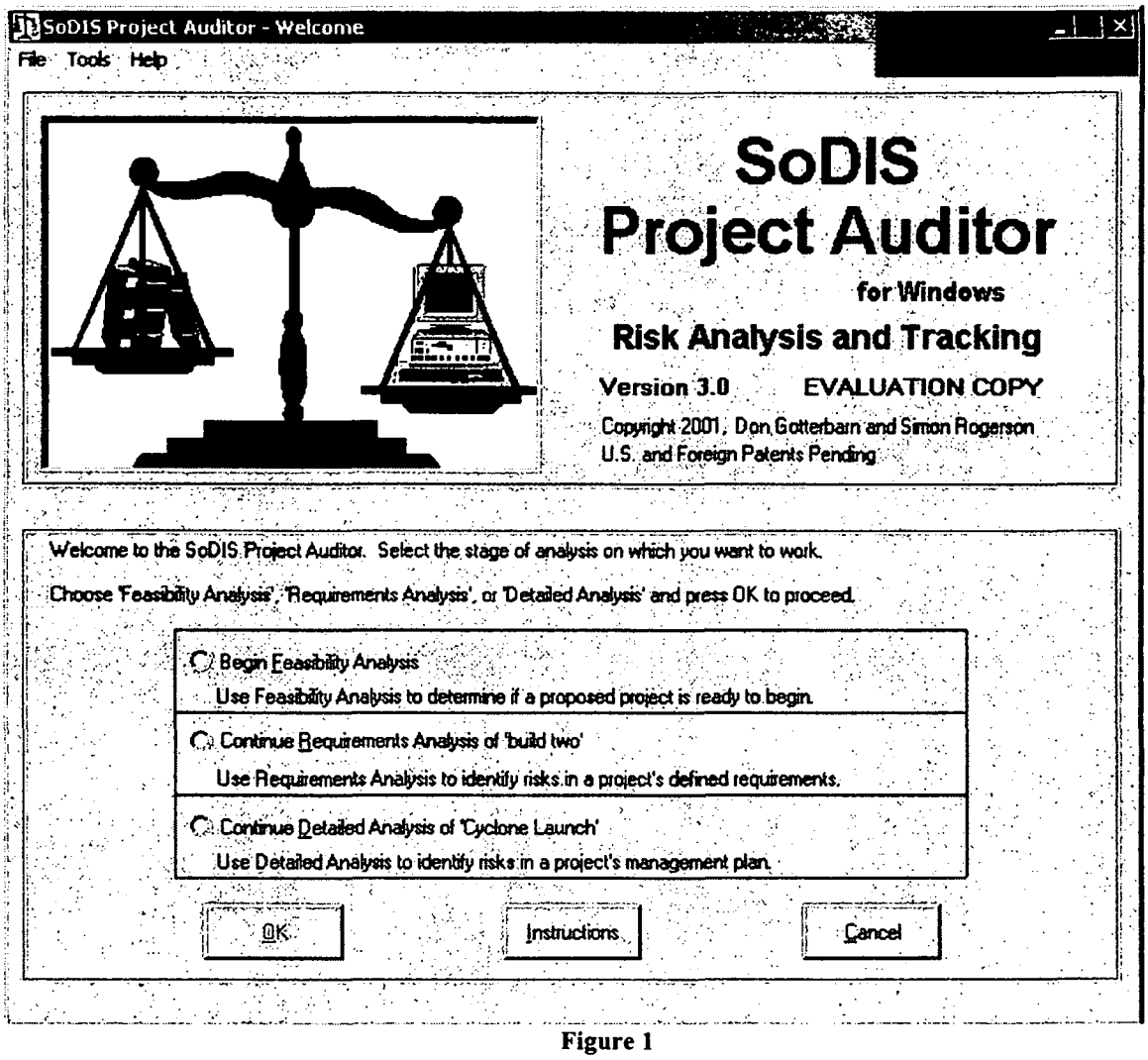


Figure 1

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.

SoDIS 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.

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 identification 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 to address the identified concern.

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.

Identification of Stakeholders

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.

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 that will now include the new stakeholder.

Rogerson & Gotterbarn [1997] proposed a method to help identify stakeholders 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, personal liberty, free speech, and property. How can these rules be used to identify stakeholders?

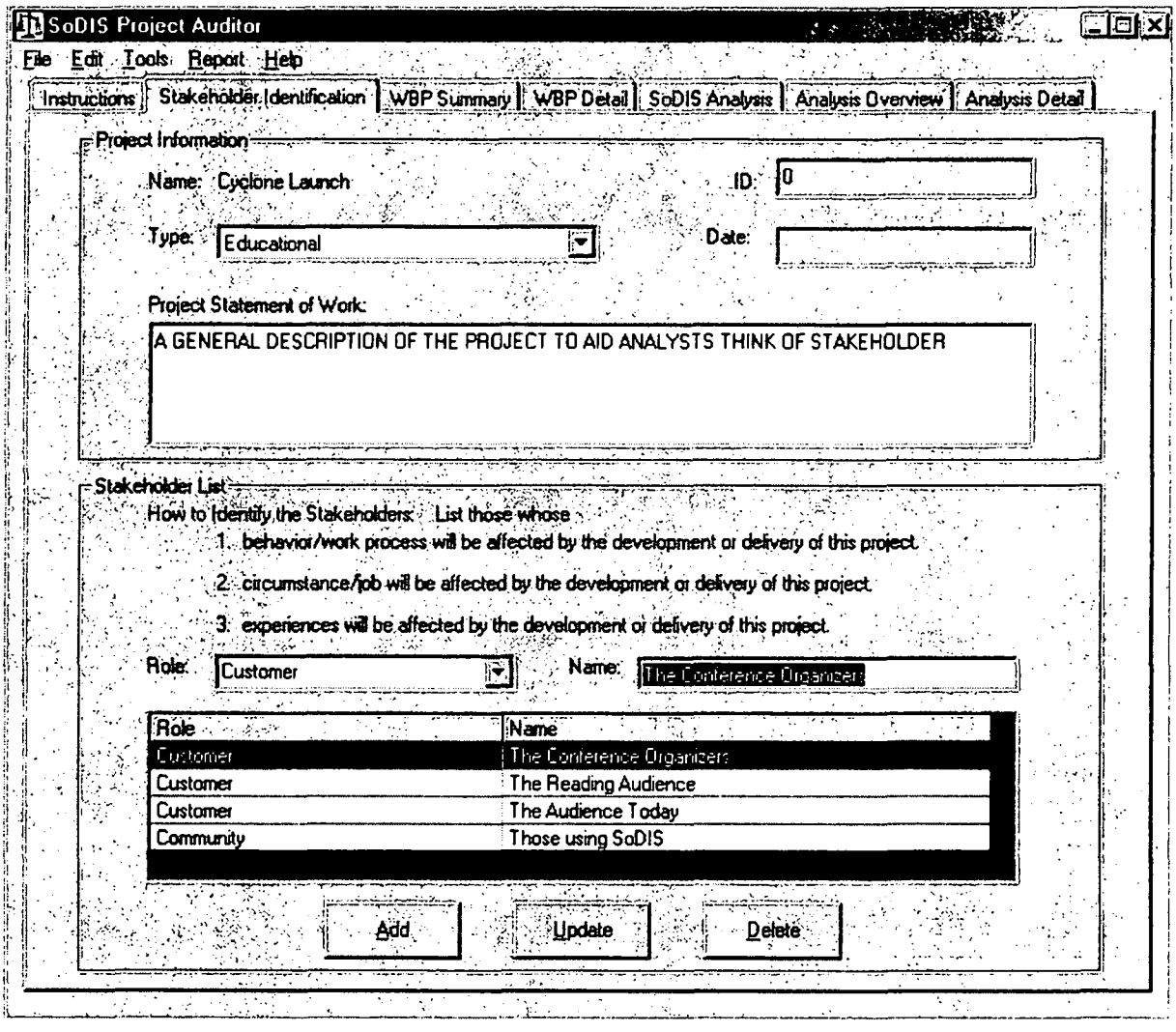


Figure 2. Stakeholder Identification

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

Req\Stakeholder	Customer	developer	user	Community	Additional stakeholders...
Requirement 1	N	N	N	N	
Requirement 2	N	N	N	Y	
Requirement 3	Y	N	Y	Y	

This process can be 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 groups, and society as potential stakeholders.

Identification of Tasks

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 3).

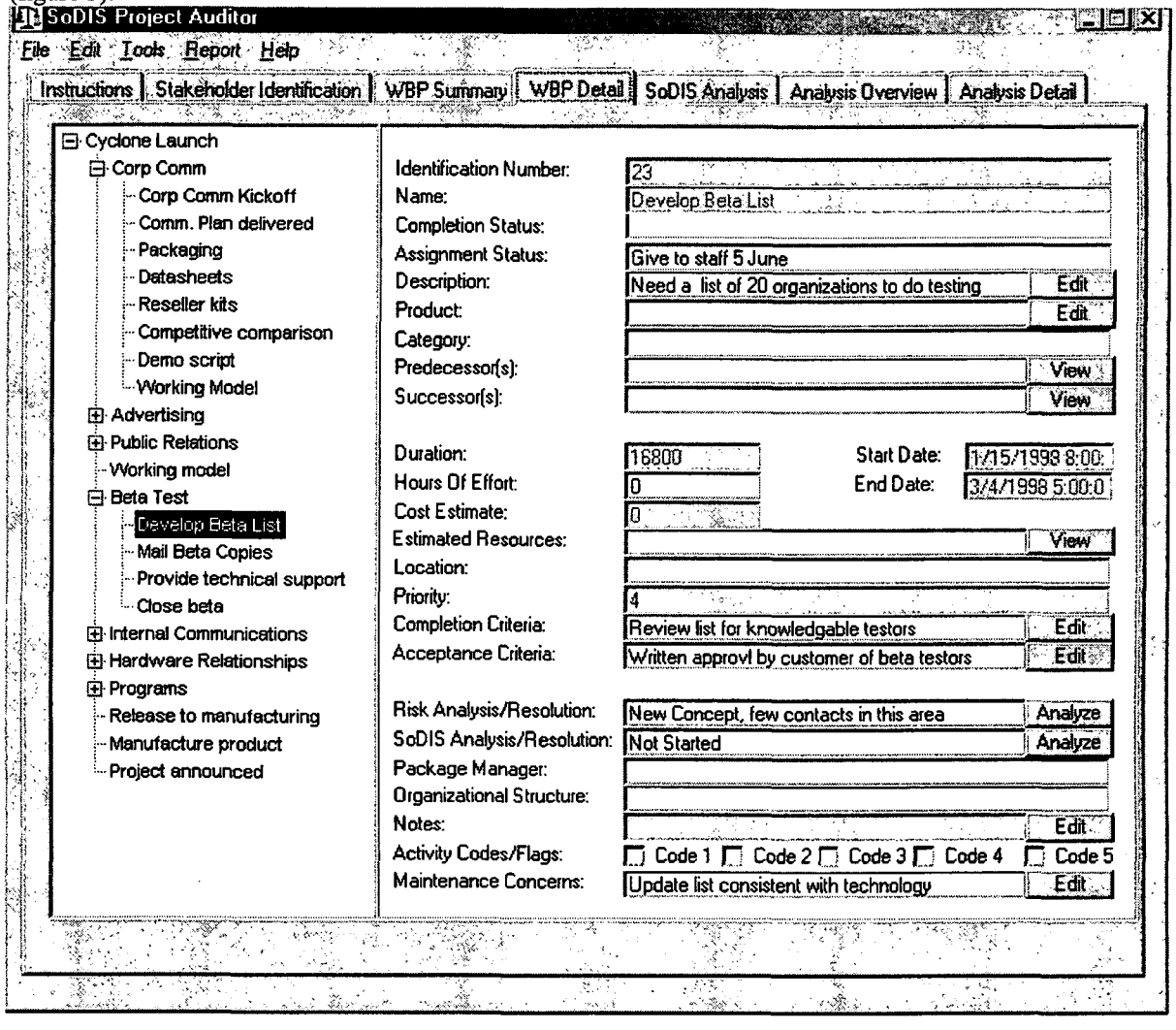


Figure 3 WBP Detail Screen;

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.

Identify Potential Ethical Issues

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. These questions are placed in the bottom frame of the SoDIS Analysis screen (figure 4).
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.

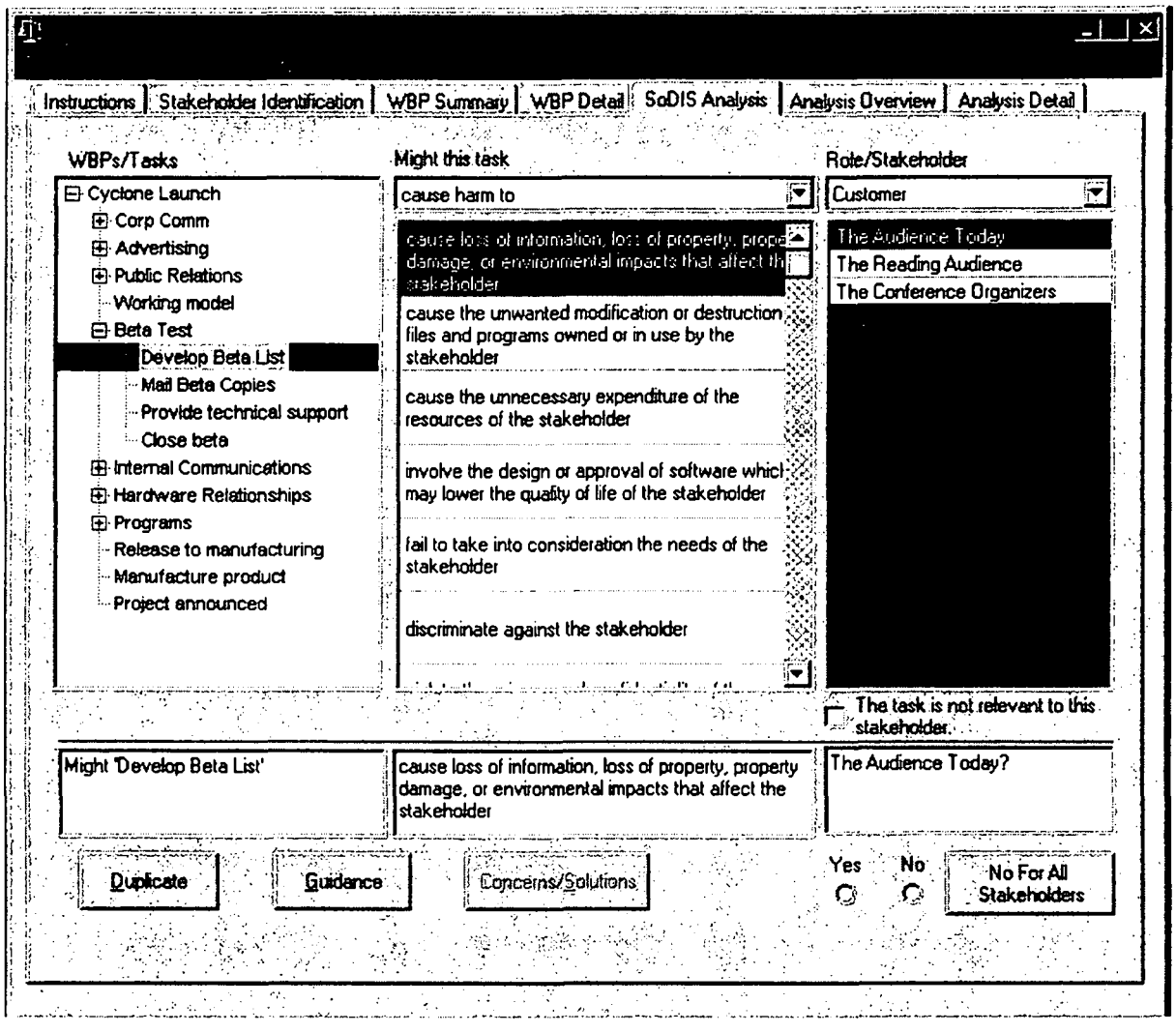


Figure 4 SoDIS Analysis screen;

When an ethical concern has been identified, the analyst gets an ethical concern form (figure 5) that asks the analyst to record their concern with the task and record a potential solution. The most critical part of this process is on this form, where the analyst is asked to assess the significance of their concern with the work breakdown package being 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.

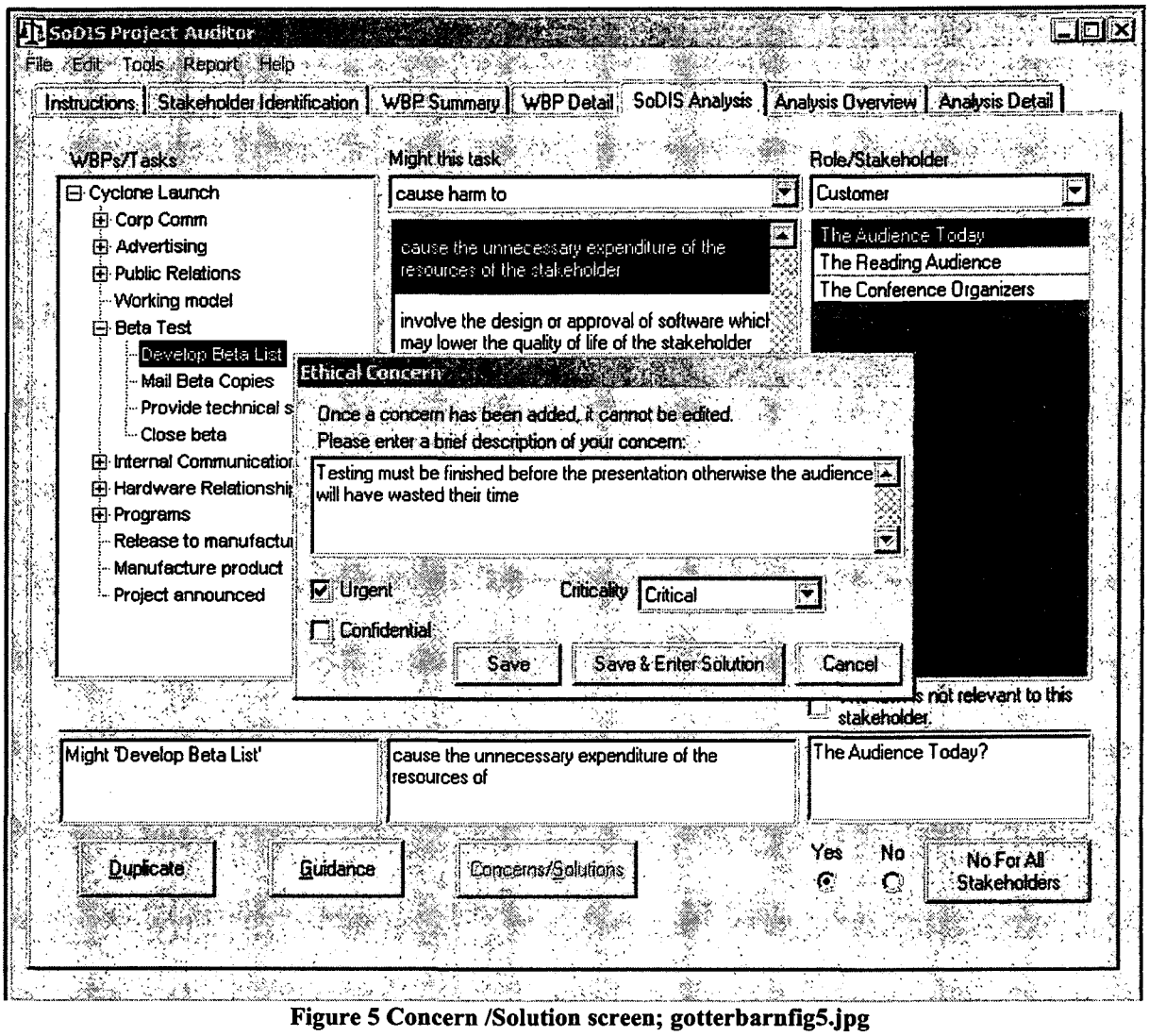


Figure 5 Concern /Solution screen; gotterbarnfig5.jpg

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.

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 SDLC.

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.

INTEGRATION OF THE SODIS AND AN SDLC

Barry Boehm's modified spiral model – the Win-Win Model (WW) – comes close to meeting 1) the stakeholder identification problem and 2) the risk limitation 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.

Although this approach is similar to Rawl's wide reflective equilibrium in deriving a coherent set of requirements through negotiation, the ethical element is missing from Boehm's method. There is no methodology to identify ethically relevant stakeholders nor is there an ethical foundation for the negotiation process.

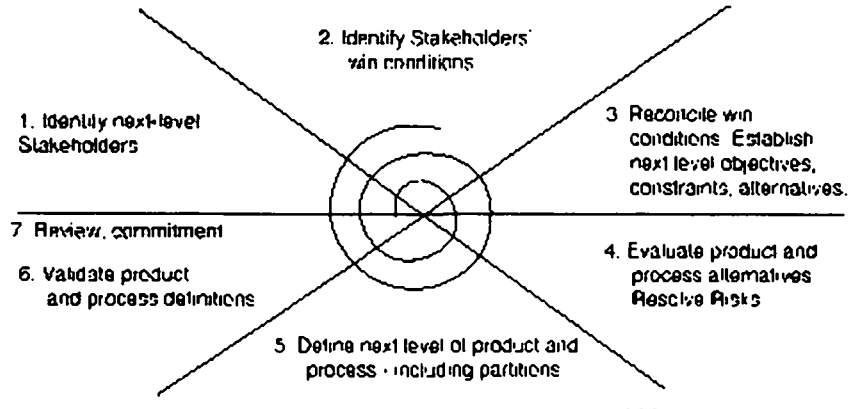
The method is also limited in that it assumes all stakeholders are equal and that they will equally be aware of and able to describe their own win conditions. The negotiation amongst stakeholders will be unjust and will likely lead to failed systems, unless, contrary to fact, each stakeholder has such 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.

This model has two strengths. First it comes closer to properly expanding stakeholder identification than other software engineering methodologies. Unlike all of the other approaches that presume the impact analysis is done as a single process, the Win-Win model is iterative requiring a re-identification of system stakeholders at each stage of the development process.

The negotiation activities include:

- Identification of system stakeholders- defined as "anyone that has a direct business interest in the system or product o be built and will be rewarded for a successful outcome of criticized if the effort fails"
- Determinations of the stakeholder's win conditions.
- Negotiation of the stakeholder's win conditions to reconcile them into a set of win-win conditions for all concerned.

Although risks and stakeholders are incorporated into the model, negotiations activity 1 does not identify a broad base of stakeholders. The sailors using the Aegis radar system and the passenger who lost their lives would not be stakeholders.



[Boehm, 1998]

The types of risks are also limited. The Project Risks include: difficulties associated with budget, schedule, personnel (staffing and organization), resources, and customer. Technical Risks include technical uncertainty, technical obsolescence, and difficulties in interfacing, maintenance, design and implementation. The Business Risks include: market risk (the product no one wants), product line skew (product does not match product line), and management risk (change in physical management or focus). These risks do not include risks to the users of the Aegis radar system.

In a further modification to his model, Boehm adds three process milestones to help mark the completion of one spiral. These three "anchor points" – life cycle objectives, life cycle architecture, and initial operational capacity – provide technical information used to make decisions about whether and how the project might proceed.

The win-win model has the ability to analyze risk and incorporate stakeholders. The inclusion of a SoDIS analysis as a starting point for the project and as the fourth anchor point then the Win-Win model would be an effective SDLC which addressed the technical and ethical risks of software development.
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:

Boehm, B.,"Using the WINWIN Spiral Model: A Case Study," **Computer** July 1998.
Buglione, L and Abran, A (1999) LIME: A Three-Dimensional Measurement Model for Life Cycle Project Management," **Proceedings of the International Workshop on Software Measurement** September 1999.
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.
McCarthy J (1996) **Dynamics of Software Development**, Microsoft Press.
Rogerson S and Gotterbarn D "The Ethics of Software Project Management", in **Ethics and Information Technology**, ed. Göran Collste, New Academic Publisher, Delhi, 1998