A business case for SoDIS

Dr Roger McHaney
Kansas State University, U.S.A.
mchaney@ksu.edu


Abstract

An applied risk analysis methodology called a software development impact statement or SoDIS is increasingly becoming accepted as a rigorous mechanism for qualitatively identifying the problems and shortfalls associated with software development projects and resulting applications. This methodology has been imbedded in a software package known by the name SoDIS Project Auditor. One of the challenges SoDIS faces is an economic justification for expenditures associated with conducting an analysis. This article examines potential pitfalls in building a traditional business case and suggests several approaches for justifying the use of SoDIS.

Keywords

Project management, risk analysis, SoDIS, software development

1. Introduction

In today's business environment, modern software deployment has emerged as more of a science and less of an art. This is not to say that innovation, creativity, and thinking-outside-the-box no longer have a place in application development. It is, however, important for these aspects to be conducted within a controlled environment where monetary expenditures and team efforts are carefully managed. More than ever, programming practices and development methodologies are being subjected to fiscal scrutiny. The resulting pressure encourages project managers and software development leaders to reduce activities viewed as having low return on investment (ROI) or high overhead costs.

While a financial eye on the software development process has encouraged positive innovations such as better CASE tools, enhanced development environments, and group support systems, the added pressure has also pushed development teams to reduce front-end efforts in order to save time and money. Often, it is the upfront planning and analysis phase of the project that suffers. This practice is short-sighted and while temporarily reducing initial costs, in the long run, a poorly planned project can balloon in scope and expenses leading to long term overruns and other more serious disasters.

Attempts to justify SoDIS and other risk analysis methodologies in a corporate environment can be difficult, especially when using techniques that focus primarily on traditional financial criteria. For tools that prevent potential problems, an economic payback may be difficult or even impossible to justify quantitatively. Therefore, it may be necessary to develop a formal business case that relies on a variety of metrics to explain why the analysis is being conducted and what intangible and long-term benefits might emerge.

2. Definition of a Business Case
A business case can be defined as a formal justification for adopting a strategic, tactical, or operational business tool or methodology. In many organizations, common practice requires a business case or financial justification be proposed and accepted by management prior to proceeding with the use of new or different methodology. When the bottom line for a proposal reveals a cost savings, reduction of labor, or other financial incentive, the business case will be generally be approved. When a financial gain cannot be demonstrated, the case will be much harder to move forward. Traditionally, financial analysis has been a mainstay of business case criteria. However, other methodologies do exist and are more relevant to justifying a risk analysis methodology.

2.1 Business Case Methodology Considerations

Business cases can be constructed in a variety of ways. Typical business cases will often have the following components:

1. Description of Proposal
2. Alternatives Description
3. Cost / Benefit Analysis
4. Risk / Sensitivity / Uncertainty Analysis
5. Acquisition and Implementation

The Description of Proposal section will contain a general write-up describing what the analyst hopes to accomplish with the proposed change. Often, this section will detail the strategic impact of the proposed implementation and provide a reasonable estimate of cost magnitudes. The Alternatives Description section provides an overview of other options that might exist in lieu of the proposed implementation. These options might include not implementing any methodology as well as several reasonable alternate options. Cost/Benefit Analysis takes a look at the financial implications of implementation versus alternative courses of action. This section of the report must be carefully developed to include long term considerations such as customer satisfaction and corporate viability. Section four of the business case, Risk/Sensitivity/Uncertainty Analysis gives management an idea of how important the proposed implementation is to the corporation and a feel for how speculative the analysis might be. Finally, Acquisition and Implementation provides a plan for acquiring and rolling out the proposed changes.

2.2 Strategic Justification

Presley, Barnett, and Liles (1995) warn that justification of particular "[enterprise] technologies can be difficult when they are judged solely on traditional economic criteria." They suggest using a "strategic justification methodology" which includes a strong consideration of "long term strategic costs and benefits in addition to more standard financial considerations".

Many tools, technologies, features and methodologies that support or undergird the software development process don't have apparent, short-term financial payouts. This makes justification difficult. Among these are:

- Quality Programs
- Formal Risk Assessment
- Work Environment Improvement Factors (ergonomics, facilities, et cetera)
- Long Term Viability Items (training, skill development, conference attendance, et cetera)
- Ethical Training and Considerations

SoDIS can be classified as a Formal Risk Assessment methodology and therefore long-term, strategic justification should be considered in development of its business case.

3. SoDIS Business Case

A SoDIS Business Case can be structured to mirror the sections described above. The following sections will give examples of the types of material and information that
would included in such a Business Case.

3.1 Description of Proposal

The use of SoDIS in a particular organization or for a particular software development project would be described in the first section of a business case.

In general, the SoDIS process conceptually modifies the environmental impact statement process used in the construction industry, and uses that technique to identify potential negative impacts of a proposed project and specify actions that will mediate those anticipated impacts. SoDIS assumes that most software projects consist of stakeholders and tasks. Stakeholders are systems, people and things affected by the system to be developed. Tasks are any activity undertaken to complete the project. The SoDIS process expands risk analysis to qualitative and professional issues as well as considering the more traditional technical aspects of risk (Gotterbarn, 2002). It also identifies how project task completion might negatively affect stakeholders then provides a framework for communicating and tracking this information to prevent anticipated problems.

The SoDIS process comprises four phases: (1) identification of the project type and relevant stakeholders, (2) identification project tasks, (3) association tasks with stakeholders using structured, ethics-based questions, and (4) documentation of the analysis, concerns, and a risk mitigation.

3.2 Alternatives Description

In development of a business case, documentation of alternative approaches to the proposed change is important. Not only does this activity help demonstrate the recommended strategy is the best course of action, it also provides documentation of the thought processes used to arrive at the recommended course of action.

Several general alternatives to the use of SoDIS exist. Among these are:

- **The Common Sense Approach.** Many software development teams rely on common sense and personal experience and judgment when developing software. This has succeeded as a short term approach for many projects but calls to mind an analogy of Russian roulette. When something goes wrong, it can lead to catastrophic system failures. The common sense approach becomes less sensible in large scale development efforts or in situations where personal injury or damage is possible. Project personal turnover, and varying levels of experience may also question the viability of relying on this approach.

- **Software Safety Analysis.** This analysis technique, typically carried out as part of the design phase in the systems development lifecycle (SDLC), provides data on potential failure of various software components and projects the impact of these failures on the overall system. While providing some overlap with a SoDIS analysis, this activity takes place later in the SDLC and offers less focus on system stakeholders, particularly those not directly involved as end-users (Fenelon & McDermid, 1993; Fenelon, McDermid, Nicholson & Pumfrey, 1994).

- **Hazard Driven Testing.** Like Software Safety Analysis, Hazard Driven Testing is conducted late in the systems development process and is used to verify that the developed system will not be susceptible to known hazards and problems (Joyce & Wong, 2003). Hazard Driven Testing is an essential activity but not a replacement for SoDIS Analysis.

- **Customized Formal Alternatives.** Other proprietary methods for identifying and mitigating risks earlier in the SDLC can be used. These methods, if used in a particular organization, can be presented as an alternative to SoDIS.

3.3 Cost / Benefit Analysis

As mentioned earlier, Presley, Barnett, and Liles (1995) warn that justification of particular "[enterprise] technologies can be difficult when they are judged solely on traditional economic criteria." The use of cost/benefit analysis to justify SoDIS use may fall victim to these concerns if cost alone is used as a criteria. When addressing cost
benefit issues, a variety of outcomes occurring while using SoDIS must be considered. Considering these outcomes, particularly those with a long-term impact on an organization, is consistent with Presley, Barnett, and Liles' (1995) suggestion to use a "strategic justification methodology" that includes evaluation of "long term strategic costs and benefits in addition to more standard financial considerations". SoDIS use may include the following outcomes:

- **Case One Scenario**: The cost of the SoDIS software and labor to run a SoDIS analysis becomes a sunk cost. No unexpected risks or feasibility concerns that weren't previously discovered are found. Benefits include analyst familiarity with the system, documentation that reasonable and state-of-the-art care was taken to discover problems prior to system implementation, facilitated dialog with stakeholders, and enhanced confidence in the development process. In this instance, costs out weigh the tangible financial benefits but long-range, strategic interests have been served.

- **Case Two Scenario**: SoDIS costs are recouped thousands of times over because major, project-altering concerns are uncovered. The analyst saves the organization money, documentation is created to protect against lawsuits, stakeholders gain respect for the developers, and catastrophe is avoided. Although money was not physically lost, the savings due to potential loss can be calculated.

Outcomes without SoDIS:

- **Case One Scenario**: No money is spent on the SoDIS software or labor to run a SoDIS analysis. No unexpected problems occur during system implementation or production use of the system. Short term money is saved. No long term risk analysis experience is gained.

- **Case Two Scenario**: No money is spent on the SoDIS software or labor to run a SoDIS analysis. Project is installed and then major problems come to light. The project loses money and customer good-will. Lawsuits are filed. Money is lost and both short-term and long-term damage results. Post Hoc cost/benefit analysis results are all too apparent.

### 3.4 Risk / Sensitivity / Uncertainty Analysis

The risk associated with using SoDIS is very low. It represents a type of analysis that at a minimum will enhance knowledge of a system under consideration for development. The information provided by SoDIS will not detract from a project but instead will make all parties more knowledgeable and provide documentation of concerns regarding a project. The risks of not using SoDIS may in fact potentially prove to be higher and result in unexpected project costs and consequences.

Since SoDIS is a relatively new product, there will be a learning curve and a need to integrate it into an organizational evaluation process. However, a growing body of academics and practitioners is emerging to provide a supportive environment for its use (Clear, McHaney, & Gotterbarn, 2004).

### 3.5 Acquisition and Implementation

Plans for acquiring SoDIS and using it within a development project would be described in the Acquisition and Implementation section of the Business Case. A commercial version of SoDIS can be obtained from Software Improvements, an Australian software development firm (www.softimp.com.au). A brochure on their Website provides a sample implementation plan (www.softimp.com.au/SoDISbrochure.pdf).

### 4. Rationale for SoDIS Use

SoDIS is intended to qualitatively identify potential harm and risks faced by individuals and systems impacted by a system implementation. The approach taken by SoDIS has the effect of broadening the stakeholder circle to include individuals, systems, and groups outside most traditional systems risk assessments. The framework for conducting a SoDIS analysis is based on ethics principles (Gotterbarn, 1993) thereby
ensuring project concerns are investigated from a variety of perspectives.

4.1 Example System Failures

The development of the SoDIS process and related software was prompted by reports of software-related incidents, reported in academic literature and the general media that resulted in unexpected consequences. Studies of these incidents revealed that many of them could have been anticipated and prevented had an appropriate analysis been conducted. High profile examples include:

- Therac-25 Incident
  - Killed at least one and injured others with too much radiation during cancer treatment (Levenson & Turner, 1993);
- Under treated patients;
  - 1000 Cancer patients given too little radiation during treatments in England (Arlidge 1992);
- Confidential medical records inadvertently shared (Gable, 2001);
- IBM System advises the release of a felon incorrectly. Later this felon commits a murder (Sprague, 2001);
- Bank of New York overdraft of $32 Billion due to software glitch (Sprague, 2001);

4.2 Challenge of Creating a Business Case for SoDIS

While the examples described in the preceding section are high profile, costly errors in software development, most project errors are quietly dealt with and don't receive publicity. The challenge with creating a business case for SoDIS is that large, costly problems may have a low rate of occurrence but a high cost when they do occur. The challenge then is to justify use of a tool which can be likened to a health insurance policy or a quality assurance program component.

5. Legal Concerns

Software development has been protected from litigation in many countries because it is a desirable industry that governments fear might suffer otherwise. Historically, software liability has not been a problem for four reasons (Amour & Humphrey, 1993). First, until recently, software was largely in the domain of expert users. Second, most early software-related products have been fairly simple. Third, software vendors market their software products under tight contracts. And fourth, the law community is only recently developing tactics to handle software-related cases.

In the United States, defective software liability has been governed according in a way that is depicted in Figure 1.
In the U.S., strict liability tort laws cover damage caused by unreasonably dangerous products. To be subject to strict liability, software must first be considered a product rather than a service or information. Although likely that at some point courts will consider software a product, case law has not supported that view. Currently, this means software is usually governed under contract law. If a contract does not exist between software developer and user or affected party, then negligence by the software developer must be proven. Under negligence, a supplier's responsibility is limited to harmful defects that could have been detected and corrected through reasonable quality control practices. If negligence is proven, the resulting awards for in U.S. courts can be very large sums of money. To avoid litigation, most software developers use warranties to contractually assure customers that their product will perform as expected. The warranty will also limit the supplier's liability in the event of nonperformance (Armour & Humphrey, 1993). Unless the court finds the developer to have used their expertise in an unconscionable fashion to construct a contract between unequal parties, the contract will generally govern any damages.

There is another situation that contracts will not necessarily cover. Amour and Humphrey (1993, p.12) start their discussion of this instance by saying: "Even if all your products are covered by iron-clad contracts, the injured person could be a third party and not be covered." The software developer has no contract with the third party and hence no contractual protection.

The current legal environment in the United States and in other countries has several implications for the software developer. First, with software's increasingly important role in all aspects of life, the developer has to be ready for the day when software is reclassified as a product and becomes subject to strict liability tort law. Second, recognition by the courts that software has an ethical obligation to an expanded set of stakeholders means that developers can no longer consider only the immediate user with whom they have a contract. Modern software development tools and techniques need to consider an expanded set of stakeholders or individuals affected by the use of software systems.

Unfortunately, pending legislation in the U.S. could dramatically and perhaps unpredictably change that country’s current legal software environment. The Uniform Computer Information Transactions Act, or UCITA, approved by the National Conference of Commissioners on Uniform State Laws (NCCUSL) in the summer of 1999, has two
primary features. "First, it validates terms held back until after payment and delivery, presented in so-called shrinkwrap or clickwrap contracts. Second, it re-characterizes software and digital content contracts as licenses of use, rather than sales of copies, raising two issues that will require a great deal of litigation to resolve" (Braucher & Henderson, 2000). The passage of this into law could further complicate the issues of whether consumer protection laws are applicable to "sales" of goods and services.

The business case for SoDIS will be well-served in recognizing these issues and proactively preparing an organization for the legal environment that is currently emerging.

6. Conclusions

The applied risk analysis methodology called a software development impact statement or SoDIS is a rigorous mechanism for qualitatively identifying potential problems impacting the well-being of an extended set of project stakeholders. This methodology has been designed to consider the consequences of various aspects of software implementation. Concerns uncovered by SoDIS are classified and managed in a way that helps protect the developer from negligence litigation and provide an understanding of potential strict liability issues. In the process of doing this, a project’s extended stakeholders are ethically represented.

This article has discussed the development of a business case for acquiring and using SoDIS. It has demonstrated the importance of considering more than the short term costs of purchasing the SoDIS software and incurring the associated labor costs. The article suggests that SoDIS should be viewed in the same way a quality assurance program or insurance policy is viewed. A relatively small investment up-front can protect an organization from potentially devastating economic consequences downstream. This manner of justification is consistent with Presley, Barnett, and Liles (1995) warning that justification of particular "[enterprise] technologies can be difficult when they are judged solely on traditional economic criteria."

Further, SoDIS qualitatively considers software impact upon an extended group of stakeholders from which software developers are not protected under contract law and warranties. By detecting potential problems prior to installation and deployment, a software development organization is ethically protecting both its extended stakeholders and its long-term future viability.

Acknowledgements

This research was supported in part with a Kansas State University Presidential Development Award (FDA #1715).

References


Copyright © 2004 McHaney, R.
The author(s) assign to NACCQ and educational non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The author(s) also grant a non-exclusive licence to NACCQ to publish this document in full on the World Wide Web (prime sites and mirrors) and in printed form within the Bulletin of Applied Computing and Information Technology. Authors retain their individual intellectual property rights.

Copyright ©2004 NACCQ.
Krassie Petrova and Roger McHaney (Eds.)
An Open Access Journal, DOAJ # 11764120, (zotero)