A Risk Management Framework for Software Engineering Practice

Geoffrey G. Roy
School of Engineering Science,
Murdoch University, Perth, Australia 6150.
Email: geoff@eng.murdoch.edu.au

Abstract

Formal risk analysis and management in software engineering is still an emerging part of project management. This paper provides a brief introduction to the concepts of risk management for software development projects, and then an overview of a new risk management framework. Risk management for software projects is intended to minimize the chances of unexpected events, or more specifically to keep all possible outcomes under tight management control. Risk management is also concerned with making judgments about how risk events are to be treated, valued, compared and combined.

The ProRisk Management Framework is intended to account for a number of the key risk management principles required for managing the process of software development. It also provides a support environment to operationalize these management tasks.

1. Introduction

Risk analysis and management is, or should be, a core requirement of most project management tasks, especially in those situations where well-laid plans do not always come to fruition. Software development is one of those areas, but formal risk analysis is rarely an integrated part of the project management task. While Boehm [1] has laid the foundations and Charette [2] outlined the applications, there have been few widely developed formal risk methodologies that are tailored for the software development industry.

Risk management for software development is primarily concerned with the process and less with the final product. While software products may not meet the specified functional requirements, it is rare that they fail in use. Most often software products work *exactly* as built, much to the frustration of users of many popular software products.

The primary concern is therefore with ensuring the integrity of the software development process. If it goes well there should be minimal unforeseen negative impacts on the conduct of the development project and, by inference, on the resulting product. At very least the goal is to have all identified risk factors under effective management control.

There is still considerable confusion about the roles of quality assurance/management processes and risk management – since both are primarily concerned with process. It should be clear to a close observer that the foci of the two management tasks are diametrically opposed. Quality assurance is concerned with adopting a process that will minimize that chances that the project can deviate from the set development pathway. Risk management is concerned with identifying what might go wrong, and how these events will impact on the project. While the outcome goals are similar, the strategies to be followed are quite different. We can thus expect to see distinctive theories and models being developed and applied in each case. The role of the risk manager is much more that of a devils advocate compared with the quality manager.

Formal risk analysis is built on probability theory. Vose [3], for example, provides a detailed coverage of the subject, in the context of risk analysis, including the theoretical foundations. Most software development projects will, however, involve a (large) number of risk factors. Apart from trivially small cases, it is extremely difficult to obtain detailed (quantitative) estimates of the required probabilities. The challenge is therefore to devise useful methodologies that can be applied to practical-sized problems. These methods should impose a level of theoretical integrity that makes them to be both generally acceptable and practically useful.

In the context of software development there is a relatively short history of experience to build upon. In other, more established, design-based disciplines (especially in most of the more traditional engineering areas) there are well-established bodies of knowledge, codes of practice and professional standards. Software Engineering, as a discipline, is still coming to terms with these issues. As a result there is still a lot to learn about how the risks can be modeled and managed in software development projects.

2. Risk in Software Development

Risk is concerned with uncertainty. This naturally includes uncertainty about the occurrence of known events, but also events that are not initially identified as impacting on the project. Risk management must therefore be an evolving and learning process, adapting to new and changing knowledge as the project proceeds.

Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)
1530-0803/04 $ 20.00 © 2004 IEEE

Risk value is generally defined as the product of the impact (or effect) of a risk event and the probability of the event occurring, i.e.

$$V(A) = P(A) * C(A) \quad \text{Eq 1}$$

where P(A) is the probability that event A will occur, C(A) is the cost/impact or risk effect, and V(A) is the risk value for event A. The risk value is interpreted to be the average expected loss (in some measure) of the event occurring. This is also referred to as utility loss.

This form of risk valuation has some intuitive appeal, as the impact of a risky event will be higher if either the probability increases or the cost increases. As a result, a low cost event with a high probability can have the same risk value as a high cost event with a low probability. For commensurate risk factors this is not unreasonable. Commensurate risk factors that those that:

- are measured on the same cost/value scale, and
- are not extreme events, i.e. risk the factors do not have substantially different (e.g. many orders of magnitude) probabilities and/or costs.

For a small number of possible events this product model is simple enough and can provide a useful basis for ranking and comparing risky events. As the complexity increases it is necessary to consider:

- the independence, or otherwise, of the risk events, and
- the need to combine risk events to enable effective management, assessment and control.

It may be necessary to elicit, from the project management team, a number of probability values or even probability distributions. It may also be necessary to estimate conditional probabilities amongst selected events, given that some will not be totally independent of others. These tasks are complex, if not intractable, for practical-sized projects.

If some assumptions about the conditional probabilities amongst the risk factors can be made then it is possible to suggest some workable strategies. For example, the combined impact of a set (cluster) of risk factors may be estimated by:

$$V_I = \sum_{i \in I} w_i \cdot V_i \quad \text{Eq 2}$$

for events that are substantially independent, or

$$V_I = \prod_{i \in I} w_i \cdot V_i \quad \text{Eq 3}$$

for events that are essentially mutually exclusive events, or

$$V_I = \max_{i \in I} \{w_i \cdot V_i\} \quad \text{Eq 4}$$

for a fuzzy model of the union of the event outcomes (i.e. Murphy's Law: if anything can go wrong the event with the largest risk value will be the one!). The weighting coefficient (w_i) is taken as the risk effect of the event occurring and hence scales the impact of the event into an appropriate utility measure.

Various interpretations of these model primitives have been used. For example: an additive formulation is used within the SERIM [4] model, and one based on a product form is used within the risk extensions to the COCOMO II (see Madachy [5]) estimation framework.

3. Risk Models for Software Development

A taxonomy of software project risks has been developed at the Software Engineering Institute (Higuera and Haimes [6]). This taxonomy defines a tree-structured hierarchy of risk areas, appropriately classified to define clusters of risk factors. The top three levels are shown in Table 1. The 4th level (Level 3) corresponds to sets of questions that probe for potential risks, and thus defines the risk factors for the risk model.

The contributions of each cluster of risk factors to the project risk can be estimated by accumulating the risk values up the tree. The relative contribution of each risk factor, and cluster, is described by weighting factors that measure the risk effect of each. These weighting coefficients have been estimated from a number of case studies, and could provide a starting point for a project-wide risk model.

Table 1: The SEI Taxonomy (upper levels)

Root Node [Level 0]	[Level 1]	[Level 2]
Project Risk	Software Development Risk	Requirements
		Design
		Code and Unit Test
		Integration and Test
		Engineering Specialties
	Development Environment	Development Process
		Development System
		Management Process
		Management Methods
		Work Environment
	Program Constraints	Resources
		Contract
		Program Interfaces

The SEI taxonomy is comprehensive, though still quite informal. It requires a detailed analysis of the project and its operating organization, and from this process much of the required detailed knowledge and appreciation of the risks are elicited and exposed. It thus forms the basis for a comprehensive awareness-raising exercise. This is, of course, an essential part of the risk management process.

Karolak [4] has proposed a more formal modeling framework, similar to the SEI taxonomy but providing a more quantitative, and more detailed approach. In particular this (SERIM) model offers the risk manager four interconnected risk trees based on 81 risk factors. These are referred to as risk perspectives and the top three levels are shown in Table 2.

As with the SEI model, the Level 3 nodes (not shown here) are expressed as a set of questions that probe the risk factors for their likelihood of occurrence. The risk factors are shared in each perspective, with different sets of weighting factors applied in each case.

The SERIM model is more complex in that is it really four models (i.e. four root nodes) with each one corresponding to a different risk perspective. These might also be aligned with different stakeholders that are associated with the project (e.g. different parts of the management team).

The SERIM model requires the user (risk manager) to provide estimates of the likelihood for each risk event. The SERIM modeling tool then computes the combined risk values at each level of the risk tree using a simple additive formulation like Equ 1. The risk manager is then able to compare the risk value contributions from each part of the risk tree and within each risk perspective to identify the key risk factors.

Table 2: The SERIM Risk Perspectives

Root Node [Level 0]	[Level 1]	[Level 2]
Risk Elements	Technical	(a) Organization
		(b) Estimation
		(c) Monitoring
		(d) Development Methodology
		(e) Tools
		(f) Risk Culture
		(g) Usability
		(h) Correctness
		(i) Reliability
		(j) Personnel
	Cost	(a) to (j) as above
	Schedule	(a) to (j) as above
Risk Categories	Process	(a) to (j) as above
	Product	(a) to (j) as above
Development Phases	Pre-requirements	(a) to (j) as above
	Requirements	(a) to (j) as above
	Design	(a) to (j) as above
	Coding	(a) to (j) as above
	Testing	(a) to (j) as above
	Delivery and Maintenance	(a) to (j) as above
Risk Activities	Identification	(a) to (j) as above
	Strategy and Planning	(a) to (j) as above
	Assessment	(a) to (j) as above
	Mitigation and Avoidance	(a) to (j) as above
	Reporting	(a) to (j) as above
	Prediction	(a) to (j) as above

4. A Risk Management Framework

Risk management must be an integrated part of the project management framework if it is to be effective. Much has been written in this area, for example see Charette [7, 8] for a comprehensive coverage. In addition, various codes of practice offer generic guidelines, for example AS/NZS 4360:1999 [9] provides the essential guidelines to establish appropriate management practices, as shown in Figure 1.

The steps shown in Figure 1 must be wrapped in a process that appropriately identifies roles and responsibilities within the organization and the project team. The ProRisk Management Framework follows these steps with operational extensions. The goals of these extensions are intended to provide:

- A set of guidelines to assist in the analysis of a project for risk elements.
- A process by which these risk elements can be organized into groups of related risk factors and ultimately into risk perspectives that match stakeholder views.
- A model representation that enables formal risk analysis to be performed using a quantitative approach while keeping the data requirements to a minimum,
- A process of analysis that assists in the identification of key risk factors, outcomes and reactions, and the creation of action plans to mitigate these risks, i.e. to target resources where the payoffs are expected to be the greatest.
- An ongoing re-assessment to ensure continuous monitoring and review of the risk elements as the project proceeds towards completion.

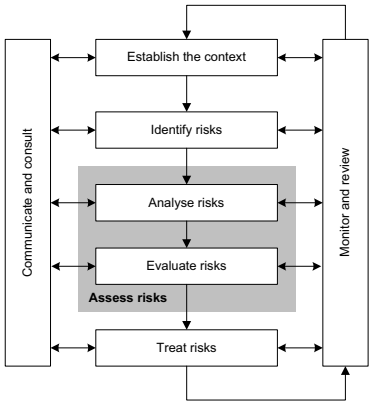

Figure 1: The AS/NZS 4350:1999 Risk Management Framework

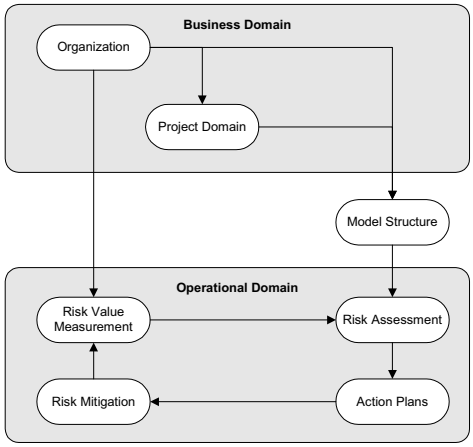

Figure 2: The ProRisk Management Framework

The proposed framework, in the context of typical software development projects, is shown in Figure 2.

This framework focuses attention on primary project components, i.e. the business domain in which the project is created, and the operational domain when the project is actually carried out.

The business domain is a necessary part of the risk management process as it provides opportunities to:

- Identify the economic environment in which the project is being undertaken, and the susceptibility of the organization to the performance of the project team and the exposure to external risk factors.
- Estimate the knowledge and experience of the organization for the project, and the level of confidence that the project can be successfully concluded.

These two factors define the components of the framework that allow a formal model of the risks to be described.

The operational domain contains the formal modeling aspects of the project:

- Undertake the necessary measurement of risk values as guided by organizational views and policies.
- Complete detailed assessments to identify the key risk factors within the assumed modeling framework.
- Identify and describe the action plans aimed at reducing key risk values.
- Implement these plans and then re-assess the effected risk factors.
- Wrap these steps in a continuous cyclic process that must be applied for the duration of the project.

The ProRisk Management Framework is also built on an hierarchical model structure along the lines as described in the SEI taxonomy of risk [6, 10] and Karolak [4], and requires the following activities:

4.1. Stakeholder Identification

The stakeholders in a project are those individuals, groups of people, or organizations that benefit from the outcome of the project (and by inference suffer a loss if something goes wrong). A stakeholder may have one or more perspectives, and these perspectives may have their own measurement units. To be effective in representing the interests of stakeholders the methodology must be able to support both a range of stakeholders and different ways of valuing their interests.

4.2. Risk Factor Identification

The initial stakeholder identification will assist in risk factor identification; but more information is needed. Stakeholders, by themselves, are probably unable to produce a complete set of risk factors that describe their own perspective of interest. The aim is to identify all the things that can go wrong, and this is the task of the risk manager. It is possible to start from scratch, perhaps using a brainstorming session or a Delphi process [11, 12], to elicit a complete set of risk factors for the stakeholders and the project. This is likely to be a time consuming process, but may be justified in some circumstances.

Given that a number of well-developed sets of risk factors have been published it is reasonable to start with one of these models and modify it, as necessary, to suit the particular project needs. The SEI taxonomy [10], for example, offers a rather complete (perhaps definitive) set of risk factors for the software development project. This taxonomy has some 194 risk factors classified in a four level hierarchy. As defined, the SEI taxonomy does not offer explicit stakeholder perspectives; these may be implied in the way the risk factors are clustered to form the intermediate nodes in the risk tree.

Collofello [13] offers a simplified version of the SEI model, reducing its size to some 36 risk factors. While this model has been proposed for classroom use, it may still provide (with some minor changes) a suitable starting point for a first-time user for practical software development projects. There are also other models that are promoted by proprietary risk management tools/methodologies that may also provide useful starting points.

An important aspect of risk factor identification is to minimise, as far as possible, the dependencies between risk factors. The assumptions made (e.g. Eqs 2, 3 and 4) may ignore these dependencies. Ideally risk factors should be orthogonal to each other; that is, each should attempt to measure a unique risk element within the project. Where this is not possible special care must be exercise to choose appropriate modeling tools that allow conditional probabilities to be adequately represented. The ProRisk Framework does not support these types of models.

4.3. Risk Tree Model Construction

Building a complete risk tree model is an iterative, and learning, process requiring the involvement of all stakeholders and the risk management team who will be responsible for the overall integrity of the model. From the initial list of risk factors (perhaps from an existing template) the first task is to develop the risk clusters to collect together sets of commensurate risk factors that align with the stakeholder perspectives. All risk factors in a cluster should relate to a similar aspect of the project. This is just common sense good management practice. A risk cluster can be represented by the simplest of tree models.

In practice, clusters can be formed from shared subsets of risk factors like that shown in Figure 3. The separate clusters may be measured in the same, or different, units as required.

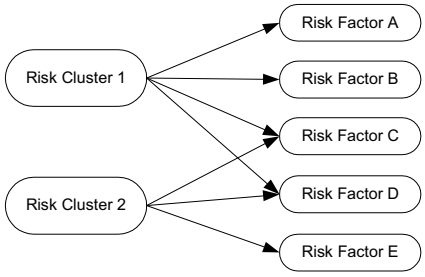

Figure 3: Risk Clusters

The complete risk tree will be formed by further aggregations of the clusters until each perspective has a single root node representing the whole project (at least as defined for the respective stakeholder). A complete project may be represented by one or more risk trees, or perspectives to match stakeholder requirements. For example, one perspective may represent the total cost risk and the other the total schedule risk.

An example risk tree model is shown in Figure 4 and Figure 5. This model has two risk perspectives: cost and schedule, based on a shared set of risk factors (the leaf nodes). The descriptions of the risk factors are given in Table 3. This model is based on a model developed and implemented by Strategic Systems [14] for a trial of the ProRisk Framework.

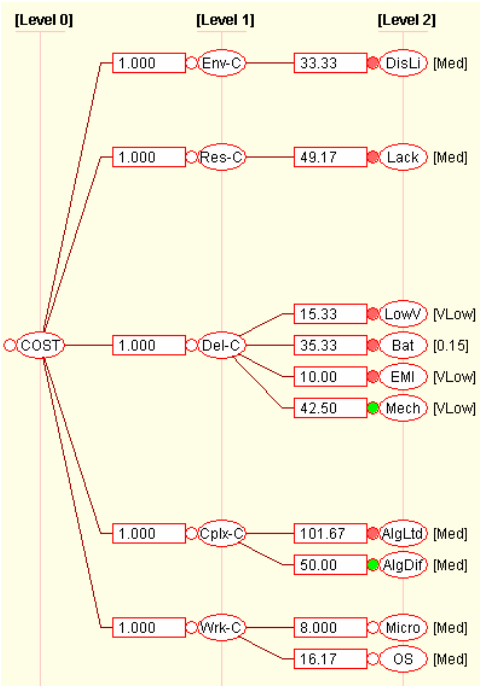

Figure 4: Sample Risk Tree Model: Cost Perspective

4.4. Calibrating the Model

The primary risk value calculation requires values for the probability that a risk event will occur, and its cost/impact (in appropriate measures) for each risk perspective to which it belongs. Obtaining these values is not trivial and may well test the limits to which a quantitative model can be developed.

The calibration of the model begins with the estimation of the weighting factors (for example, as included in Eqs 2, 3 and 4). Each risk perspective must be treated independently, for example:

- *Cost Perspective*: the weighting factor will be a dollar value; perhaps taken as the average cost the event will add to the project, if the event occurs.
- *Schedule Perspective*: the weighting factor will be a time value (days, weeks, etc); taken as the average delay the event will cause to the completion of the project.

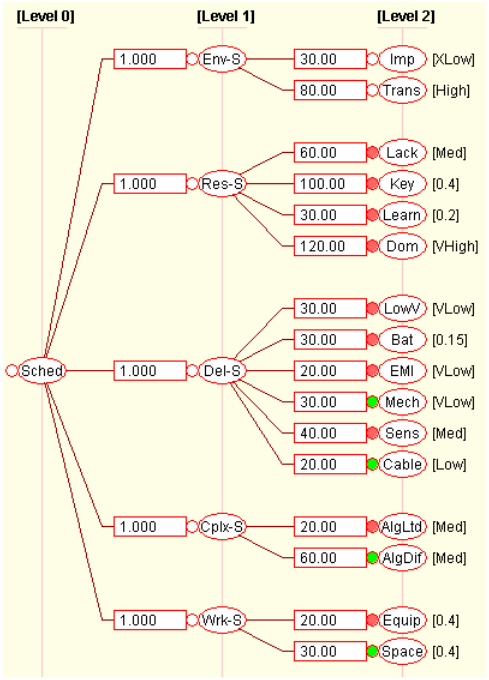

Figure 5: Sample Risk Tree Model: Schedule Perspective

Other risk perspectives (e.g. like quality and reputation) must also be measured if they are to be used. The process for these more qualitative perspectives is not as straightforward, but approximations can be made using simple normalized scales to define relative values. Within the ProRisk Framework these perspectives can co-exist in the overall model.

The use of simple normalized values for risk is not uncommon. This approach is used in the SERIM Model [4] and for the sets of weighting values that have been derived from surveys undertaken from within the SEI [6]. In these cases the risk values provide a relative value estimate that can be used for comparing risk values (identifying the top ten, for example) without requiring a absolute value interpretation. So far only "average" values for the weighting coefficients have been referred to. In reality the cost/impact of a risk event is likely to be uncertain, perhaps following a probability distribution. In some circumstances it may possible to provide the probability data for a risk event. For example, if the cost impact of an event can be described by a worst-case, a most-likely case and a best-case value. In this way an attempt can be made to describe a probability distribution, albeit from just three data values (e.g. using triangular or BetaPERT distributions).

Table 3: The Risk Factors

Group: Complexity
AlgDif: Real-time processing algorithm in software difficult to develop - several iterations, increased complexity, etc|
AlgLtd: Real-time processing algorithm limited - may need different algorithms for different configurations, etc

Group: Work Environment
OS: Operating System to be used - development environment, compilers, delivery system costs
Micro: Microcontroller to be used - test environment and production costs...
Space: Infrastructure (network, phones, etc) and lack of available space for new team members
Equip: Suitable test equipment

Group: Deliverables
LowV: Sourcing and delivery of suitably low voltage embedded micro-controller for the on-board real-time processing
Cable: High tension fibre-optic cabling delivered on time (late May)
Mech: Mechanical engineering - all offsite - delays, quality check, etc
Bat: Supply and transportation of lithium batteries
Sens: Sensors - source and supply of suitable components
EMI: EMI/EMC testing (need to send off-site, will they be ready when we need it, etc)

Group: Environmental
Trans: Transport of Lithium batteries - held in customs, time for shipping, etc
Imp: Impact on environment - objection to system install due to potential for negative environmental impact
DisLi: Disposal of Lithium batteries

Group: Resources
Dom: Availability of an suitable Domain Expert throughout the entire project
Lack: Resources and the lack thereof
Learn: Learning curve for new resources
Key: Assignment of key staff to the project in the timeframe required

A risk model developed with this level of detail can provide the risk manager with a much richer view of the project risks. The risk manager will be able to observe the likely range of risk values (rather than just having an average), and thus be able to judge whether the extreme values require further attention (for mitigation purposes). It is, however, unlikely that this level of information will be available for most software development projects.

4.5. Estimating the Risk Event Probabilities

This is the final step in the model building phase, and necessary before the model can be used for anything useful. Each risk factor must be assessed for its likelihood of occurrence and a probability allocated. Generally these are represented by values on a 0 to 1 scale.

Adjective scale values using a Likert scale (e.g. XLow, VLow, Low, Med, High, VHigh, XHigh) can allow users to think in more descriptive terms rather than numeric values (which is accepted as being difficult for some people). In this latter case there must be an underlying risk-value function that maps the chosen scale to numeric values. The default mapping will probably be linear (over the chosen range of values), but nonlinear mappings can be developed where sufficient knowledge and experience is available.

4.6. Computing Combined Risk Values

Within the ProRisk Management Framework a number of combination models are provided:

- *Summation*: combines risk values by summing values at each risk cluster according to Eq 2.
- *Product*: combines risk values by multiplying risk values at the lowest level cluster (Eq 3), then summing at higher levels (Eq 2).
- *Max*: based on Murphy's Law (Eq 4) that says that if anything can go wrong the event with the largest risk value will occur, i.e. taking the most pessimistic outcome. This corresponds to a fuzzy logic approach [15] for the union of the possible events.

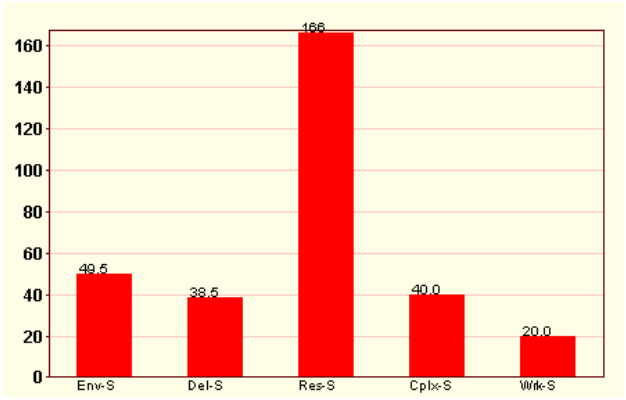

Figure 6: Exploring the Risk Tree

Once built and calibrated (with organizational or project specific values) the model can be used to assist with the identification of the key risk factors. Figure 6 shows the view of the model within the risk tree; in this case, from the root node for the schedule perspective. Similar graphs are available to explore the children nodes, their children and so on. Naturally, risk factors can also be ranked according to their contributions to the various risk perspectives built into the model. Figure 7 shows the top 10 set of risk factors in order of their contribution to the project as a whole.

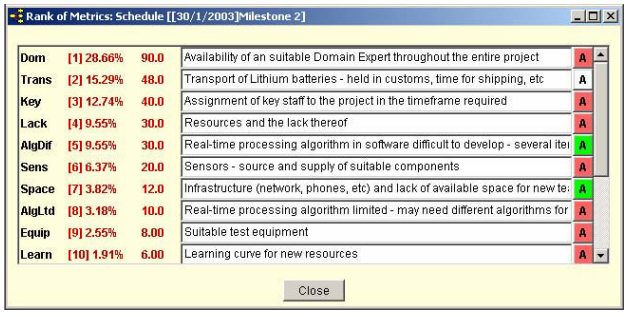

Figure 7: The Top 10 Risk Factors

4.7. Developing Action Plans

Within the ProRisk Management Framework, the Action Plan provides support for the documentation, management and re-evaluation of risk events during the progress of the project. An action plan requires:

- A delegation of responsibility, i.e. who is responsible for carrying it out.
- A description of what will be done to reduce the impact of a risk factor.
- An allocation of resources to enable the work to be done.
- A time scale describing when the work will be done.

The completion (or anticipated completion) of an action plan should lead to a re-evaluation of a risk factor, normally a change (hopefully a reduction) in the estimated probability for the risk event, or a revised estimate of the risk impacts. With a revised probability level inserted into the model, the effects of the change will be immediately observable. The gains in risk reduction must be valued against the costs involved, perhaps by using a risk-reduction leverage value (Boehm [1]) that may indicate whether the gains in risk reduction are justified by the costs incurred.

4.8. Monitoring the Progress

As a project proceeds the risk values will change from time to time. These changes will come from natural stochastic events as well as management interventions resulting from mitigation actions. To enable appropriate management this temporal behaviour must be recorded so that the progress of the risk properties can be monitored.

The ProRisk Management Framework allows a risk model to evolve over time, maintaining a snapshot of the state of the model at each time epoch (that should align with normal project milestones), allowing the temporal changes in the model to be recorded and monitored. Figure 8 shows a summary of part of the cost perspective over three project milestones.

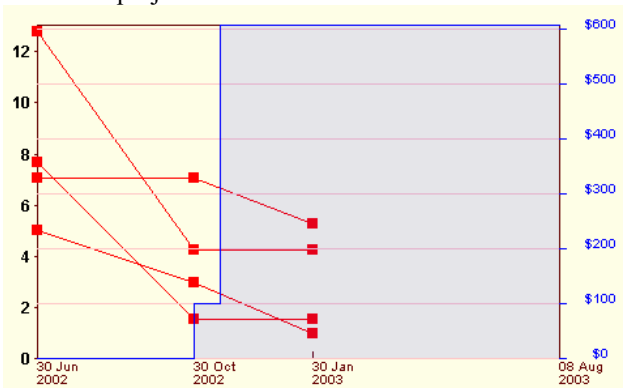

Figure 8: Risk Monitoring with Mitigations Costs

In this figure we can observe the changing risk values (following the implementation of Action plans and subsequent reviews of risk values. We also can observe the cumulative costs incurred, i.e. the mitigation costs.

4.9. Operationalizing the Framework

To be useful the framework must be operationalized in a management support tool. There are a number of risk management tools available, but few that have a particular focus on software development projects and fewer match the requirements of the framework described here. To be effective:

- The tool should be transparent and its internal structure clearly exposed to the users (risk manager) and peers (stakeholders).
- The underlying modelling framework should be as simple as possible and not clouded by complex computations and/or data requirements.
- All the parameters should be clearly exposed to the user, with options to adjust to suit project and organization needs.
- All assumptions must be made explicit.
- The user/risk manager should be able to tailor and specialize the model to suit local contexts and project-specific needs.
- The tool should support "what if" analyses to allow the user to experiment with the model, firstly to prove the integrity of the model, and secondly to provide a means of exploring its behaviour under parameter variations.
- The tool should facilitate a continuous monitoring and documentation of the risk properties, and provide support for risk mitigation and management on a continuing basis.

5. Summary

Software development projects are particularly demanding for risk analysis. They encompass a wide range of risk factors across a number of stakeholder-determined perspectives. Most of the currently available tools are either not scalable to large problems, hide many of the key assumptions built into the modeling framework, or are very demanding on the quantity of information required to set up and calibrate the models.

The proposed framework can be readily applied to both small scale and quite complex projects, with manageable levels of data requirements. It is also a relatively open system, allowing users to develop their own specialized models, and to calibrate these to suit the needs of the operating organization and the domain of the project. Models can be defined and refined as experience grows.

The ProRisk Management Framework requires a detailed analysis of the organization and project domains to develop a complete set of risk factors and to ensure they are appropriately organized to reflect all the stakeholders and the various risk perspectives that are required. Building a complete risk model from scratch can be a

substantial task, but it is also possible to use, or build on, one of the published models (templates). In this way considerable experience is obtained at minimal cost. This would be the normal starting point. ProRisk is provided with a number of these templates.

The framework covers the complete life cycle of the project and provides support to allow the risk analysis to run in parallel with the conventional project management activities.

The integrity of the ProRisk models can only be assured if the key underlying assumptions are maintained, these include:

- the risk factors are largely independent or mutually exclusive,
- the appropriate methods are used for computing combined risk values,
- the risk perspectives are constructed from commensurate risk values, and
- extreme risk events are excluded from the model.

Given these requirements then the ProRisk Management Framework offers a range of new capabilities for the risk management of software development projects. The framework has been implemented with the ProRisk support tool [16]. A demonstration version of ProRisk can be found at http://eng-sun3.murdoch.edu.au/~geoff

6. References

[1] B. W. Boehm, *Tutorial: Software Risk Management*: IEEE Computer Society, 1989.

[2] R. N. Charette, *Software Engineering Risk Analysis and Management*. New York: McGraw-Hill, 1989.

[3] D. Vose, *Risk Analysis: A quantitative guide*, 2 ed: John Wiley & Sons, 2001.

[4] D. W. Karolak, *Software Engineering Risk Management*: IEEE Computer Society, 1997.

[5] R. J. Madachy, "Heuristic Risk Assessment Using Cost Factors," *IEEE Software*, vol. 14, pp. 51-59, 1997.

[6] R. P. Higuera and Y. Y. Haimes, "Software Risk Management," Software Engineering Institute, Carnegie Mellon University CMU/SEI-96-TR-012, June 1996.

[7] R. N. Charette, *Software Engineering Risk Analysis and Management*, vol. 1: Cutter Consortium/ITABHI Corporation, 2001.

[8] R. N. Charette, *Applications Strategies for Risk Analysis*, vol. 1: Cutter Consortium/ITABHI Corporation, 2001.

[9] Standards Australia, *AS/NZS 4360:1999, Risk Management*, 1999.

[10] M. J. Carr, S. L. Konda, I. Monarch, F. C. Ulrich, and C. F. Walker, "Taxonomy-Based Risk Identification," Software Engineering Institute, Carnegie Mellon University CMU/SEI-93-TR-6, June 1993.

[11] N. Dalkey and O. Helmer, "An Experimental Application of the Delphi Method to the Use Of Experts," *Management Science*, vol. 9, pp. 458-467, 1963.

[12] H. A. Linstone and M. Turoff, *The Delphi Method: Techniques and Applications*: Addison-Wesley, 1975.

[13] J. Collofello, "Questions List for Software Risk Identification in the Classroom," Arizona State University Software Risk Management Home Page, http://www.eas.asu.edu/~riskmgmt, 2000.

[14] Strategic Systems, "Project Risk Case Study," http://www.ss.com.au, 2002.

[15] R. E. Bellman and L. A. Zadeh, "Decision-making in a fuzzy environment," *Management Science*, vol. 17, pp. 141-164, 1970.

[16] G. G. Roy, *ProRisk User Guide*: Murdoch University, School of Engineering Science, 2003.