

Managing Software Project Risk

Robert Armstrong, Gillian Adens

Tassc Limited
www.tassc-solutions.com

First Published: March 2001
Last Updated: January 2008

Copyright 2001-2010, Tassc Limited. All Rights Reserved.

The process of developing a software solution based on plans and schedules containing estimates, assumptions and other uncertainties, (not to mention the people factor) is a risky business. All software projects are exposed to some degree of risk, but it's how we manage risk that can make the difference between delivering a successful solution and counting the cost of a failed project.

In this paper we examine common risks facing software developers and managers, and present a case study with practical tips and guidelines on reducing and controlling software project risk.

Risk Management

*don't try to avoid
risk, just be
prepared*

Risk is commonly perceived as being exposed to the possibility of adverse circumstances, and for software projects the result is invariably commercial loss. But risk is also part of economic enterprise and the source of entrepreneurial profit. Software organisations need to take risks to be competitive and development projects need to take risks to innovate. In the pursuit of some advantage, risk is unavoidable and in some circumstances may even be desirable.

Risk management is about reducing or eliminating the negative impact of potential problems. Rather than passively reacting to events, those responsible for managing risk need to be proactive, vigorous and diligent, both in advance and during the software development process.

The first step in effective risk management is to be aware of potential problems that might adversely affect your software project. To do that you need to establish a mechanism for identifying and handling risks and then actively search out potential risks, [1].

Discovering Project Risks

carry out a risk survey to identify risk as early as possible

Prepare a risk survey asking management, developers, other development teams, customers and any other project stakeholders to describe the most critical issues facing the development project and to state the importance of each. The objective of this exercise is to produce a comprehensive statement of all potential issues and risks facing the project.

A risk survey can take the form of a questionnaire or may be done by interviewing participants. The method chosen should reflect the circumstances of the organisation and must not be overly cumbersome or bureaucratic.

This exercise is ideally carried out early in the project lifecycle - but it's never too late to start. No matter what stage in the development process, a risk survey will yield some value.

Common areas of potential risk exposure on software projects include:

- clarity, completeness and stability of requirements
- reliability and accuracy of project estimates
- availability, skill level and quantity of development resources
- experience and quality of project management
- stability, reliability and availability of appropriate technology
- suitability and reliability of development tools
- dependency on contractors, vendors, suppliers and service providers
- availability, suitability and reliability of external components and services

An important point here is to be reasonable and practical. It is unrealistic to expect to identify every possible problem in advance. As long as those risks that could have significant impact are considered and tackled, then the overall risk to the project is reduced.

Categorising Risks

consider market and technical risk

The next job is to collate and categorise the risks.

A risk survey typically raises a wide variety of potential project issues that need to be organised into categories to be profiled and analysed for opportunities to reduce risk. The results obtained from the risk survey will vary according to the perspective of those individuals contributing as well as being influenced by the stage in the development lifecycle.

Common problems typically reported in a software project risk survey include:

- poor definition and frequent changes to requirements
- unreliable estimates, unrealistic schedules and inadequate project tracking
- high staff turnover and shortage of skilled staff
- lack of project standards and processes
- lack of design and inadequate documentation
- inadequate testing and quality procedures

These types of problems increase the probability of the project failing to meet performance, schedule or cost targets and are categorised as technical risk. Technical risks at the closing and post development phases of a project will include system reliability, availability, response time, support and training.

Areas of risk not often addressed involve the probability of the delivered solution failing to meet the needs of the market. Because market risk is often considered less objective and quantifiable than technical risk, most organisations, and project teams in particular, tend to concentrate on technical risk.

In some ways market risk is a more important cause of project failure. Technical issues can be resolved (or ignored) before the software product is released. In contrast, market problems will only appear after the product is shipped. The consequences of inadequate market research, unclear customer desires and the underestimated effect of competing products can turn a technical success into a commercial failure.

Understanding the Business Context

establish a risk threshold

While there are many risks inherent in the process of developing software solutions, there are many others, perhaps more critical, that relate to the organisation and the market being addressed. Technical and market risk can only be effectively addressed within the context of business priorities – those strategic business success factors critical to the organisation. You cannot sensibly judge the priority or impact of risks without an understanding of the business objectives and priorities.

Each organisation will have a unique set of business priorities, which will often include:

- increase customer service and satisfaction
- improve delivery time to market
- reduce dependency on contractors
- improve staff skill levels
- reduce development, maintenance, support and production costs
- improve cost estimating and project planning
- benefit from reuse by buying in component solutions

In developing a project risk profile the main consideration is anticipating those problems that are most likely to occur and those that have the most damaging potential. The risk survey will generate many potential problems covering a wide range of severity levels, and it is therefore important to be practical in this exercise. Establish a risk threshold - a measure of severity of the problem - that suits the project and the organisation, and only consider those risks above that threshold.

Prioritising and Quantifying Risks

prioritise the potential problems and create a project risk profile

When all project risks have been identified, captured and categorised, the next task is to prioritise the potential problems and create a project risk profile.

Each risk needs to be evaluated and assigned a priority based on the probability of its occurrence, and its potential impact on the success of the project (both technical and commercial). Those risks combining a high probability of occurrence and a high impact are of highest priority.

This is perhaps the most difficult part of risk management. Assigning appropriate priorities requires not only an understanding of technical risk, but also having the knowledge and awareness of the strategic business direction and market trends.

Developing Contingency Plans

*contingency
planning is a
form of
insurance*

The next stage in managing project risk is to develop contingency plans – a course of action for each potential problem identified in the project risk profile.

Carefully examine each problem and develop one or more alternative solutions. Where appropriate, involve the development team, others within the organisation and customers in solving each problem.

Contingency planning is a form of insurance and comes with a price. Anticipating problems, determining the probability of problems occurring, evaluating the potential impact and preparing solutions in advance requires considerable effort. The trade-off is that a problem that has been identified, with a solution developed in advance, is far simpler and cheaper to resolve than one that occurs unexpectedly.

The very act of carrying out this type of exercise in itself reduces project risk! Not only have the risks been identified and resolutions considered, the communication of the results raises broad awareness of potential problems early on and instils confidence in the professionalism of the development organisation, [2]. This creates an environment and a culture that is prepared, and where the probability and impact of unexpected problems is reduced. But don't be heavy handed. Putting too many risk monitoring and contingency plans in place can create a significant project overhead that can slow progress on the project, which in turn increases the risk of late delivery.

Reducing Risk

*typically
contingency
plans are treated
as passive
documents*

As well as developing specific contingencies, is it good risk management practice to address broader and more fundamental project issues.

Reduce market risk by encouraging frequent customer contact during development. Get back to the customer often with a demonstrable product. Ideally involve a customer representative in the development team. The more direct involvement the better.

Reduce technical and market risk by building prototypes – technical prototypes proving the software architecture, communication mechanisms or functionality in key or high risk areas, and market prototypes to test usability and user acceptance.

Reduce overall project risk by establishing an incremental development process. A series of frequent incremental deliveries increases customer confidence in the likely success of the project, as well as testing the development team's ability to deliver, verifying designs, and proving the development process.

Typically contingency plans are treated as passive documents, waiting for some problem to occur before being referred to and activated. An important point about contingency plans is that they can contain active contingencies. This is where the risk manager is proactive and executes an active contingency before the anticipated problem even occurs. For example, if a high-speed data link is critical to a project, the risk manager can appoint two or more potential suppliers, primary and backup. In the event that the primary supplier announces a delay, the project can switch to a backup that is already primed.

Getting into the Habit

risks should be analysed and tracked regularly throughout the software development process

As a project progresses through the lifecycle, it is possible to anticipate and encounter additional problems as new information becomes available. For example, a major design decision is found to be incorrect. Understanding the impact of this type of problem and determining the best solution will involve some backtracking and will introduce schedule delay.

Likewise, prototypes and early development iterations will test and verify the software architecture and should actively eliminate potential risks to a project.

The project risk profile (and project estimates) will need to be revised to reflect new information. Like software estimating, risk management should not be a one-off up-front activity. To be most effective, risks should be analysed and tracked regularly throughout the software development process.

Managing Scope Creep – A Case Study

typically software projects are not well defined at the outset

One of the biggest risks and most demanding challenges facing software project managers is that of establishing the requirements for a software project.

Without a clear understanding of business objectives and requirements it is quite likely that a project will fail to deliver a solution that meets business needs. Furthermore, without a clear understanding of the requirements, how can software professionals hope to scope their development efforts and produce realistic project plans and schedules? Many systems integrators are asked to tender on the basis of an extremely sketchy RFT (Request For Tender). Is it therefore any wonder that we see so much bad publicity in trade and general press about large consulting organisations failing to deliver software on time or within budget?

A modern buzzword in the software industry is 'scope creep'. This term refers to the phenomenon of ever increasing demands for additional functionality during the development of the system. All too often requirements are not stable. There can be two distinct reasons for this – lack of scope definition and lack of scope stability.

Scope Definition and Stability

the management challenge is to measure and account for the impact of change

Often an abstract statement of requirements is prepared which is open to interpretation, and no clear boundary is established as to what functionality lies within or outside the scope of the project. Even after commercial agreements are in place, it is not uncommon to experience disagreement during the detailed scoping exercise.

Modern software development lifecycles are iterative and actively encourage the participation of end users or user sponsors. It is only reasonable to expect that intelligent and motivated individuals from the user community will bring fresh ideas to the project. They may suggest additional requirements or influence and change priorities. This is positive in that it refines our understanding of business needs and allows us to ultimately build a better system. However the management challenge is to control the stability of the project and measure and account for the impact of change requests.

Managing Scope Creep

deploy a formal estimation technique to measure scope creep

The most practical and powerful weapon that can be used to guard against the 'scope creep' risk is to deploy a formal estimation technique. It is our experience that the vast majority of organisations, even large and well-respected software development specialists, do not currently apply a well-defined science to their project estimates. Sadly, gut feel rules the day. Why is this a problem? Well, estimates based on gut feel are difficult to justify externally and do not produce consistent or repeatable results. Furthermore they often rely on one or two experts within an organisation and therefore cannot easily become part of the corporate standard processes or best practice.

Modern projects using object, component or web technology can apply ObjectMetric estimation theory, [3]. Projects using more traditional technology could benefit from COCOMO (Constructive Cost Model) or FPA (Function Point Analysis), [4]. The chosen approach is secondary. The critical issue here is to establish a consistent, well-documented, repeatable process for project estimating, planning and scheduling. The advantage is that justifiable estimates can be produced based on the agreed scope of the project.

Where the scope definition is abstract, the estimates can be refined during the detailed scoping exercise. Where 'scope creep' is as a result of new ideas and requirements, estimates can be produced to understand the impact of these changes. Therefore if and when the scope of the project expands, the customer or end user will understand and accept the impact on project schedules and costs. The use of a formal estimation technique introduces a professional mechanism for managing the risk associated with lack of clarity and stability in early requirement statements.

Understanding the Level of Definition in the Statement of Requirements

estimates of effort should be adjusted to account for the level of scope definition

Use case modelling which is now established as part of the Unified Modeling Language, [5], has become a standard technique for extracting business knowledge at an appropriate level of detail to clarify requirements and facilitate estimates for project planning and scheduling purposes.

A good estimation technique will allow you to simply and easily extract scope information from requirements statements, and using metric data, predict the effort involved in the software engineering discipline. With iterative development the definition of requirements will remain at analysis level until such time as that portion of the scope is considered a priority for development. Therefore in selecting an estimation technique it is important that estimates can be produced from analysis information and do not rely on completion of detailed design, or lines of code.

Estimates of effort should be adjusted to account for the level of definition in the statement of requirements. At the early stages in a project, the requirements will be purely conceptual and very high level. For example, we may be asked to tender to develop a system for customer contact management which will allow users to set up and maintain customer information, details of personnel within customer accounts and record contact histories.

With just a little customer involvement it would be possible to explore this very abstract statement and start requirements analysis of the customer's needs. This might result in a candidate list of user functions (or use cases) and a candidate list of business objects (classes). A metric can be applied to each use case and each class in turn to estimate the effort involved. This can be refined with additional information such as the size, complexity and level of reuse involved in each task.

Moving from Abstract to Concrete Scope Definition

as the population of software artifacts increases the level of contingency tends to decrease

After an initial statement of requirements has been agreed, the iterative software development process can then start. During analysis and design the analysis level software artifacts are translated into concrete classes and use cases for implementation. The diversity and number of resulting implementation classes is generally a function of the implementation technology, software architecture and deployment strategy.

No matter what the eventual solution, the process of refining scope understanding from a conceptual analysis view to concrete artifacts is a journey of discovery. At each stage the population of software artifacts tends to increase and the level of contingency that needs to be built into the estimates tends to decrease. The important thing is to know where you are in this process, to apply appropriate metrics and to adjust the level of scope contingency in estimates. The further into a project, the more definite and confident you will be about the effort involved, Figure 1

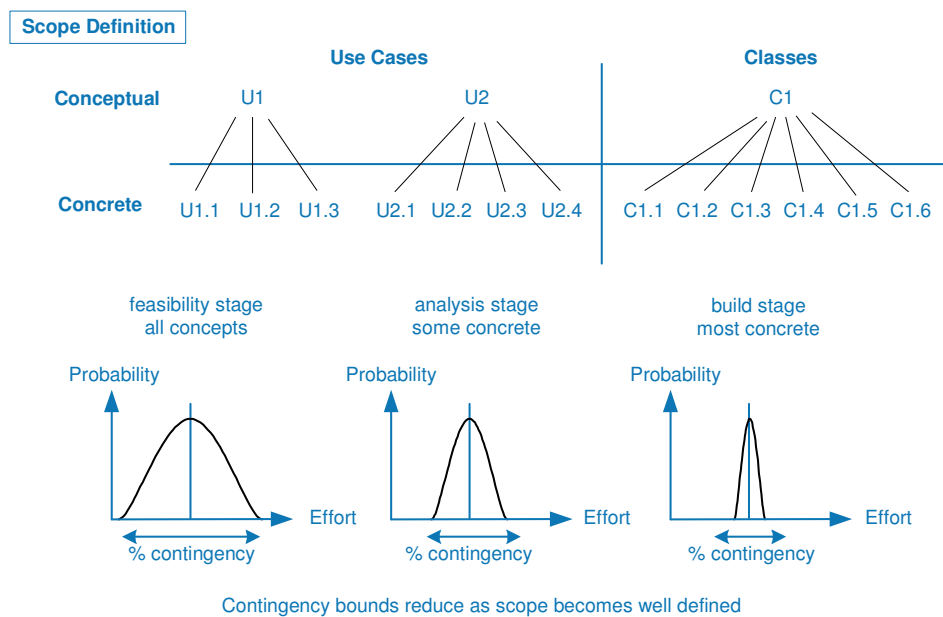


Figure 1: Scope Population Increase

The People Factor

adding developers will decrease duration until the optimum point is reached

Given an estimate of effort involved in software development it is possible to extrapolate to duration and cost. The calculation of project duration is not as simple as it may at first seem. In fact, most commercially available project schedulers do not accurately predict duration given effort and resources. They tend to take a very naive approach and simply divide effort by people to produce duration. This takes no account of skill levels and no account of the communication and management overheads that are inevitable in teamwork.

Where there is a certain amount of effort required, clearly a team of developers will complete the task more quickly than a single individual. This is true to a point, however as the team size is increased, the communication levels also increase and channels of communication become more formal. Therefore adding additional developers will decrease duration until the inter-personal communication and management overhead outweighs the advantage of further partitioning the work, as is famously documented in the Mythical Man Month, [6]. By applying an overhead for each team member (a higher overhead being applied for novices) a graph can be plotted to illustrate the optimum number of resources for a particular project, Figure 2.

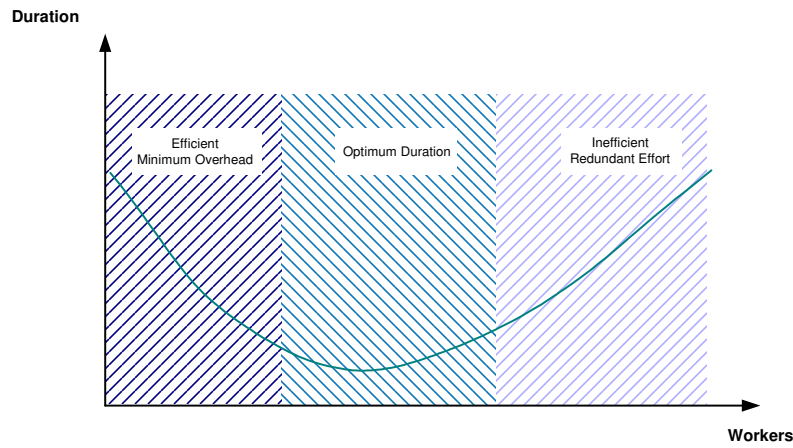


Figure 2: Optimum Resource Graph

Building in Contingency for Technical Risk

assessing risks is often subjective

To document technical risks, a simple mechanism is to produce a table enumerating each risk. Against each risk, a probability, impact and contingency can be recorded.

Judging the probability of risks and their impact is to some extent subjective. However, with a clear understanding of business objectives and priorities it is generally possible to agree a level for each. From a practical stance a sensible label should be assigned to each level and the range of levels should be sufficient to differentiate, without resulting in endless discussion. Typically a practical set of levels might be as follows: Probability levels – remote, unlikely, 50/50, possible and probable. Impact levels – minor, low, high, severe and critical, Figure 3.

Risk	Probability	Impact	Priority	Contingency
R1	Possible	Low	Low	————
R2	Possible	Critical	High	Plan A
R3	Remote	Severe	Low	————
R4	Probable	Severe	High	Plan B
R5	Possible	Critical	High	Plan C

Figure 3: Risk Profile Table

profiling risk against thresholds

Having built a risk profile table, the information can be plotted on a graph to analyse the overall project risk and illustrate those risks that are a priority for contingency plans. Probability is plotted against impact. The top right hand quadrant, Figure 4, is clearly the priority for contingency planning. Where the axes are set depends on the thresholds that the organisation wishes to define and the levels of risk that the project can sensibly tolerate.

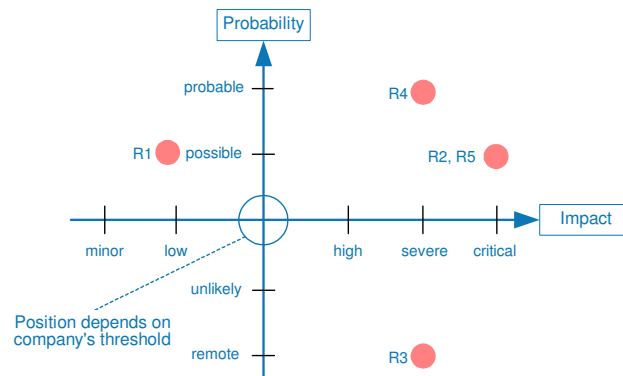


Figure 4: Risk Profile Graph

Scope contingency can establish bounds on an estimate that defines a normal distribution curve where the project effort can be potentially increased or decreased depending on what scope is uncovered during the software development process. Technical risks on the other hand, can only adversely affect estimates. This tends to skew the estimates by a factor based on the number of risks, their probability and potential impact. Risks firing can only delay the successful completion of a project.

The duration of the project can be plotted to take account of both scope risk as well as technical risk. The level of technical risk exposure is analysed to calculate a risk contingency quotient that can be applied to potentially elongate project duration, Figure 5.

The resulting graph can be segmented into 4 areas:

- The 'impossible' zone – given the amount of scope, the project simply cannot be completed this quickly.
- The 'optimistic' zone – the earliest possible delivery dates assuming the project runs smoothly and that scope is well defined and stable.
- The 'probable' zone – the most likely delivery dates for the project.
- The 'conservative' zone – pessimistically the project may take this long if scope creep is experienced and risks fire.

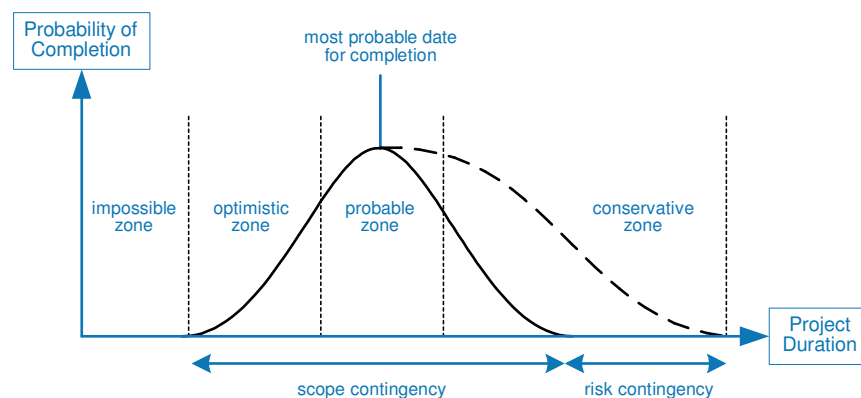


Figure 5: Risk Contingency Graph

References

- [1] Bennatan, E.M., Software Project Management, McGraw-Hill, 1994.
- [2] Gilb, T., Principles of Software Engineering Management, Addison-Wesley, 1988.
- [3] The ObjectMetrix Estimation Process, Tassc, <http://www.tassc-solutions.com>
- [4] Garmus, D., Herron, D., Measuring The Software Process, Prentice-Hall, 1996.
- [5] UML Specification, Object Management Group, <http://www.omg.com>
- [6] Brooks, F. P., The Mythical Man-Month, Addison-Wesley, 1995.