

## The Role of Risk in a Modern Software Development Process

Gillian Adens

Tassc Limited  
[www.tassc-solutions.com](http://www.tassc-solutions.com)

First Published: February 2004

Copyright 2004, Tassc Limited. All Rights Reserved.

*A modern software development process is typically characterised by its focus on user requirements, iterative development and software architecture. Often the critical role of risk analysis and risk mitigation is overlooked.*

*This paper examines the role risk plays in software development and the significant impact it can have on a project's chance of success. It describes how a change from a waterfall to iterative development process can have a dramatic effect on the risk profile of a software project. Furthermore it postulates that a risk driven scheduling policy leads to more accurate and confident project plans and schedules.*

### Traditional Waterfall Development

---

*traditional  
software  
development is  
waterfall in  
nature*

The traditional software development process is waterfall in nature. The requirements are gathered, documented and signed off by the customer. Then these requirements can be analysed to produce an idealised software model. This model is refined during design to incorporate the constraints of the implementation technology. The programming can then commence and the software is integrated and finally tested for release.

*a well-organised  
and regimented  
approach*

The waterfall software development process is a well-organised and regimented approach to software development. The advantages are clear – each step in the process takes its inputs from the previous step and performs well-defined tasks to translate them into the desired outputs.

*each member of  
the software  
development  
team can  
specialise*

Each member of the software development team can specialise and acquire finely tuned skills in their own aspects of the process. Business analysts can specialise in understanding and documenting user requirements. Systems analysts will display good powers of abstraction and strong modelling skills. Designers will have detailed knowledge of the software architecture and implementation technologies. Programmers can concentrate on code generation and integration. Testers can develop test plans and assure quality in the resulting system.

*relatively easy to plan and track progress*

Project managers can sit back and observe – the waterfall process has clear steps and deliverables, which make it relatively easy to plan and subsequently track progress.

*notorious for delivering systems that are late, over budget, and fail to meet user requirements*

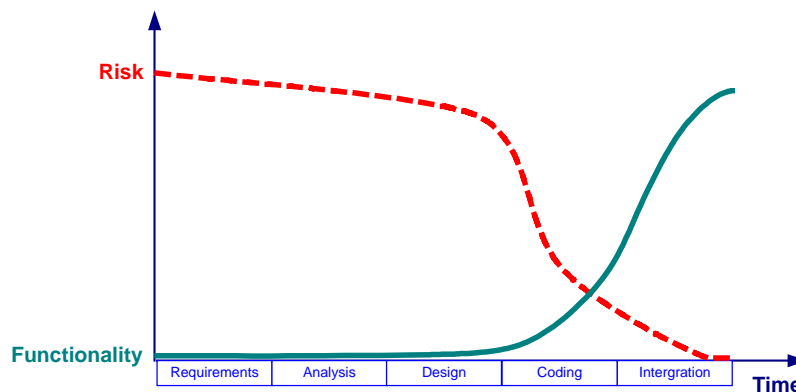
Unfortunately this is not the whole story. The waterfall process is notorious for delivering systems that are late, over budget and fail to meet user requirements.

Why is this? A waterfall process assumes that each stage can be fully defined without the need for feedback from subsequent stages. The implication is that requirements can be defined up-front completely and unambiguously and that designs can be perfected prior to coding.

In fact, requirements evolve over time and software engineers may misinterpret user needs if they do not have enough domain experience. Furthermore, if all the analysis and design is completed before any coding commences, there is a real possibility that problems found during implementation or integration may necessitate a complete rethink of the design.

*risk remains high throughout the project lifecycle*

The basic problem is that in a waterfall project risk remains high throughout the majority of the development lifecycle. It may only be at the coding or integration stage that a problem is uncovered and this can have a major impact on all of the work undertaken so far. All too often this results in major delays or even outright cancellation of the project.



**Waterfall Project Profile**

*the risk profile of a waterfall project is a major project management issue*

The above graph illustrates a typical waterfall software development approach. The x-axis shows time while the y-axis plots the amount of working software and the level of project risk. Tangible working software is almost the last stage in a waterfall process. Risk remains high until at least one full development cycle is complete – and some of the code has been built, integrated and tested.

This is a major project management issue, especially since software projects are becoming ever larger and more complex. Coupled with the introduction of new technologies such as object-oriented methodologies, UML, client-server and web architectures and new programming languages, we have a recipe for potential disaster. Such projects inevitably have a high level of technical risk at the outset, and if a waterfall development approach is taken, this risk remains high for a significant proportion of the project.

*at the half way stage there is no tangible working software and risk exposure is still high*

For example, in a two-year waterfall project, it is not unreasonable that after one whole year there is as yet no working software whatsoever. There will certainly be lots of requirements carefully documented, and lots of diagrams illustrating the analysis and design models. However, if the project is perceived as running late or over budget at this half way stage, what would be the likely view of senior project managers and the Board of directors?

The lack of tangible deliverables is not necessarily a major issue in its own right – after all a waterfall approach was chosen and presumably agreed by these same managers. However, combine this with an extremely high level of risk exposure and senior management rightly starts to feel a little less comfortable.

Perhaps the project is given a couple of months grace, but kept under careful ongoing scrutiny. This extra attention will no doubt spur the project into implementation of some ‘quick win’ end user features. Suddenly we reach the point when the risks are most likely to be activated. Risks start to fire and the project is further delayed. It is at this point that senior managers are likely to question the viability of the project. Stress levels increase all round, and a vicious circle has begun.

## Modern Iterative Development

*most companies have adopted a phased or iterative lifecycle*

Few companies continue to persevere with a waterfall approach. Most have adopted a phased or iterative lifecycle. An iterative approach allows the development team to take a subset of the full requirements all the way through analysis and design to implementation. The timescale for an iteration is relatively short – usually between three and six months. This permits the team to gain experience in the full lifecycle, and to uncover any potential problems much earlier.

*the development mode is highly dynamic and better suits a more hands-on team leading style*

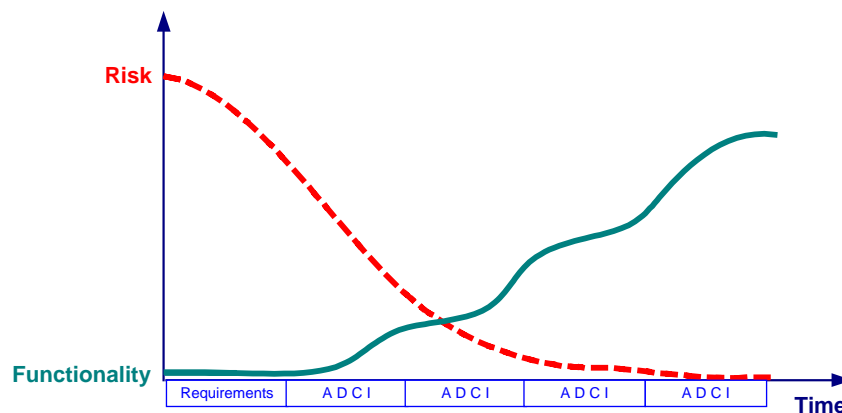
Clearly the choice to move to iterative development is not one that can be taken lightly. Such a process has a significant impact on the nature of development and requires appropriate cultural changes within an organisation. Iterative development demands smaller teams of multi-skilled individuals working more closely together. Each team member is typically involved in a wider range of activities. This allows rapid progression from one activity to the next and regular feedback and potential rework at each stage.

Furthermore the impact on project management is also significant. The development mode is highly dynamic and better suits a more hands-on team leading style. End users are typically intimately involved with the project and they are often encouraged to suggest changes to requirements. This means that project plans and schedules need to be regularly reviewed and updated.

*the adoption of iterative development requires a significant investment*

The adoption of iterative development requires a significant investment to change standards, tools, organisation and management style.

However, the dramatic impact of iterative development on the project’s risk profile, makes this change more than worthwhile.



Iterative Project Profile

*early iterations produce limited working functionality but address project risk up-front*

The above graph illustrates a typical iterative software development approach. Again the x-axis shows time while the y-axis plots the amount of working software and the level of project risk. With an iterative approach the software is developed in stages, with core functionality developed first and additional features added as the product evolves. Since the early emphasis is on putting the technical infrastructure in place, the early iterations produce only limited working functionality. However, the real benefit is that they serve to tackle the project risk much earlier in the lifecycle.

*at the half way stage an iterative project has delivered basic working functionality and significantly reduced risk exposure*

Consider the same example scenario of a two-year project. With an iterative process some basic working functionality will be in place at the end of the first year. Even more significantly, the risk profile is dramatically altered. The risks are tackled much earlier and by the end of the first year it is likely that most risks will be either dismissed or resolved. Even if the project is a little behind schedule or over budget senior project managers are unlikely to cancel this project – it has already delivered some useful functionality and there are no significant risks left so the expectation is that the remainder of the project should run relatively smoothly.

### Applying Risk Contingency to Plans

---

*risk is inherently unpredictable*

In order to produce realistic project plans and schedules, it is important to take risk into account. This is particularly crucial at the early stages of a project when risk is typically at its highest. However, risk is inherently unpredictable. Perceived risks may fire or may simply evaporate. Risks that were not even considered may suddenly emerge and have a serious impact.

*we should not be overly optimistic or pessimistic*

So how much contingency should we build into our estimates? We would be ill advised to take an overly optimistic view of our project and simply dismiss risks. Even the most experienced engineers cannot glide through a project, avoiding all risks. However, it might also be commercially unwise to go to the other extreme and expect all risks to fire. This would be a rather pessimistic view of the situation. Building in too much contingency can lead to protracted project schedules and over-budgeting. This may mean that the project no longer stacks up during cost benefit analysis, or that the company loses out to competitors in a bid situation.

*the level of contingency should reflect the level of risk exposure*

Clearly some compromise is required. The level of contingency should reflect the level of risk exposure. The number of risks that can be identified is one good indicator of risk exposure – so the more risks the higher the contingency percentage. However, this would be a rather simplistic approach. A more sophisticated model should take account of the potential impact of each risk and the likelihood of the risks firing. To derive a more accurate contingency each risk should be described in terms of its impact and probability.

*impact is the amount of damage a risk will have on the project, should it fire*

The level of impact is the amount of damage a risk will have on the duration and cost of a project, should it fire. The impact of a risk can be recorded as minor, low, high, severe or critical. These logical labels are mapped to percentages ranging from 1% to 5% (the percentages applied can be customised if required). Consequently we derive that a risk with severe impact has a potential of delaying the project by up to 4% of the overall project duration.

*probability indicates a judgement on the likelihood of the risk firing*

The probability indicates a judgement on the likelihood of the risk firing and negatively affecting the project. This can be indicated on a scale of remote, unlikely, 50/50, possible or probable. These logical labels are then mapped onto a probability percentage ranging from 15% to 85% (again this range can be customised if required). Consequently, we derive that a remote risk has a 15% probability of firing.

probability		15%	35%	50%	65%	85%	definite
		remote	unlikely	50/50	possible	probable	
impact	minor	0.15%	0.35%	0.5%	0.65%	0.85%	1%
	low	0.3%	0.7%	1%	1.3%	1.7%	2%
	high	0.45%	1.05%	1.5%	1.95%	2.55%	3%
	severe	0.6%	1.4%	2%	2.6%	3.4%	4%
	critical	0.75%	1.75%	2.5%	3.25%	4.25%	5%

**Contingency**

*the combination of probability and impact defines a risk's level of threat*

The combination of probability and impact defines a risk's level of threat to the project. This is the crucial factor in calculating an appropriate contingency for the project. Therefore a serve risk is considered to have a potential impact of 4% on the project duration. If this risk will definitely fire we would want to add a 4% contingency to our project estimates. However, risks are never definite at the outset, they only become definite if they actually fire. During risk assessment we might predict that the probability of this risk is remote and then we can calculate a more appropriate contingency by taking 15% of the 4% resulting in a 0.6% contingency. This contingency is not just based on the impact; it also takes account of probability.

*a contingency value is derived by summing the risk threats*

A contingency value for the project can be derived by taking the sum of all the risk threats – this provides a sensible level of contingency based on the combination of risk impacts and probabilities.

### Scheduling to Mitigate Risk Early

*theoretically an iterative approach will reduce risk early*

Theoretically an iterative approach to development will reduce the risk of the project much earlier than in a waterfall project. However, this statement should not be taken entirely for granted. Certainly, if risks are tackled and resolved early in a project, project plans and schedules can be produced with much more confidence. However, no project is devoid of politics, and often there is significant commercial pressure to demonstrate early progress.

*project managers are sometimes cajoled into selecting 'quick win' functionality for the early iterations*

As a result, project managers are sometimes cajoled into selecting 'quick win' functionality for the early iterations. These 'quick wins' are typically the features that avoid technical risk and lie within the team's technical comfort zone. Although this may demonstrate rapid early progress it tends to mask the real risks in the project. These risks leave the project exposed, and if they fire later in the project, may require rework of the features already delivered. Also, project plans and schedules continue to require a high degree of contingency.

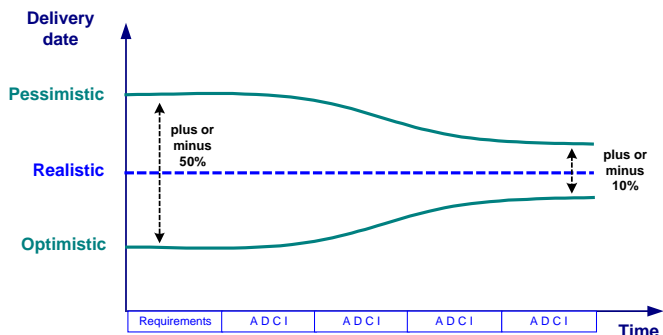
*the wise project manager will focus on the most severe risks*

Although perhaps not politically the easiest choice, the wise project manager will focus on the most severe risks as early in the project as possible. Once these serious risks are resolved the project can proceed with confidence. Furthermore, project plans and schedules can be drawn up more accurately and with minimal contingency.

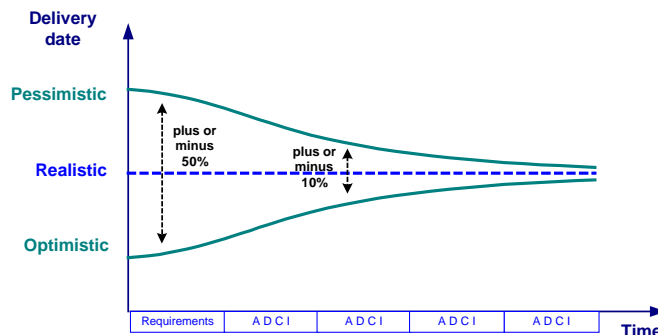
*ideally the risks will be systematically considered, addressed and resolved*

By far the best strategy is to focus on risk mitigation at the very start of the project. This immediately exposes any potential problems. At this stage there are still options and lots of time to resolve any risks that do fire. Ideally the risks will be systematically considered, addressed and resolved.

Alternatively, the risks may prove so significant that the project is no longer technically viable. In such a situation the project will need to take a completely new direction and it may mean a major rethink in terms of architecture or technology. It may even lead to the outsourcing or cancellation of the project. Although perhaps not the ideal outcome, it is much better to face the reality of this situation early when budget and time are still in reserve.



Contingency Profile - 'Quick Win' Strategy



Contingency Profile - Risk Mitigation Strategy

*uncertainty reduces throughout the lifecycle of a project*

The above picture illustrates how uncertainty and therefore contingency decreases throughout the lifecycle of a project. The scenario of the left assumes the 'quick win' strategy while the scenario on the right illustrates the preferred risk mitigation strategy.

At the start of the project the delivery date is uncertain, and a substantial contingency is built in to any estimates. As the project progresses, the iterative approach ensures that complete lifecycles are developed, each of which will address certain risks. The tendency is that as risks are tackled and resolved, the margin for error in planning will reduce and less contingency is required. This can be accelerated if risks are explicitly addressed during the first iterations.

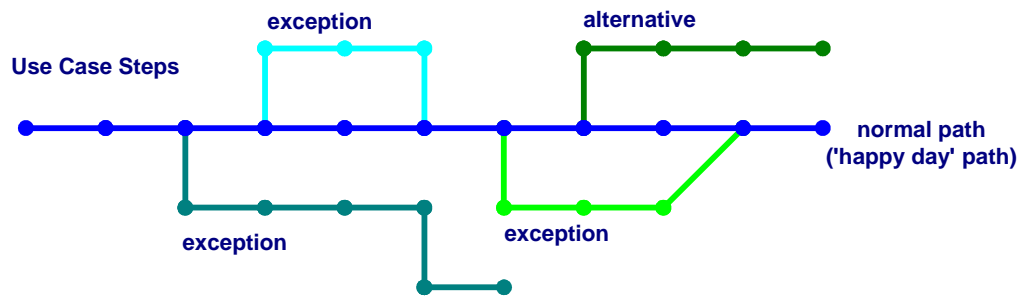
### Prioritising Features by Risk

*use cases are a great way to capture and document requirements*

Most modern software development projects define their requirements from an end user point-of-view. Use cases are a great way to capture and document requirements – these are essentially complete end-user tasks that result in clear business benefit for the user.

*the communication between the user and the system and the various steps involved*

Each use case (or user task) can then be described in terms of the communication between the user and the system and the various steps involved (often referred to as the 'flow of events'. The normal flow of events (or 'happy day' path) covers a standard successful scenario. Alternative paths are also identified to deal with unusual or exceptional circumstances. These typically require some deviation from the normal path.



*a good measure of size and complexity*

The number of steps in the normal path and the number of alternative paths for each use case is a good measure of its size and complexity respectively. This information is invaluable project management data and can greatly assist in the production of project estimates.

*traditionally prioritised to reflect business needs*

Traditionally each use case is prioritised to reflect the customer's business needs – core and essential features, important features and the extras ('nice to have' features). Next the natural action taken is to be guided exclusively by these priorities and to ensure that the first development iteration is based on some of the core and essential features.

*more effective to prioritise first by opportunity to address risks*

However, if the risk mitigation strategy is to be deployed, a different approach is required. Rather than focus on the users' priorities up-front, it is actually more effective to first focus on the areas of technical risk. Therefore each use case needs to be prioritised in terms of its opportunity to address risks. Those use cases that allow the development team to tackle serious risks (risks that have a high probability and high impact) are marked as a higher 'risk priority'.

*the skill is to identify the smallest subset of use cases that allow the largest number of risks to be addressed*

The skill is to identify the smallest subset of use cases that allow the largest number of risks to be addressed. The first iteration does not need to be feature rich, it needs to thoroughly test the software and technical architecture and exercise any known technical risks.

It may be that only the normal path of a particular use case needs to be developed, or that one of the alternative paths best tests a specific risk. A mapping exercise between use case scenarios and risks is conducted, resulting in a small highly focused technology-proving exercise for the project. Ideally this should precede the feature-focused iterations.

## Conclusion

*couple iterative development with a risk mitigation strategy to deliver on time and within budget*

In conclusion, an iterative development approach is influential in altering the risk profile of a software project in a positive way. However, this can be further enhanced by a clear strategy of risk mitigation at the very start of the project. If this approach is taken, accurate project plans and schedules can be drawn up with minimal contingency. Adopting this risk-driven approach will vastly improve the chances of complex or lengthy projects completing on time and within budget.