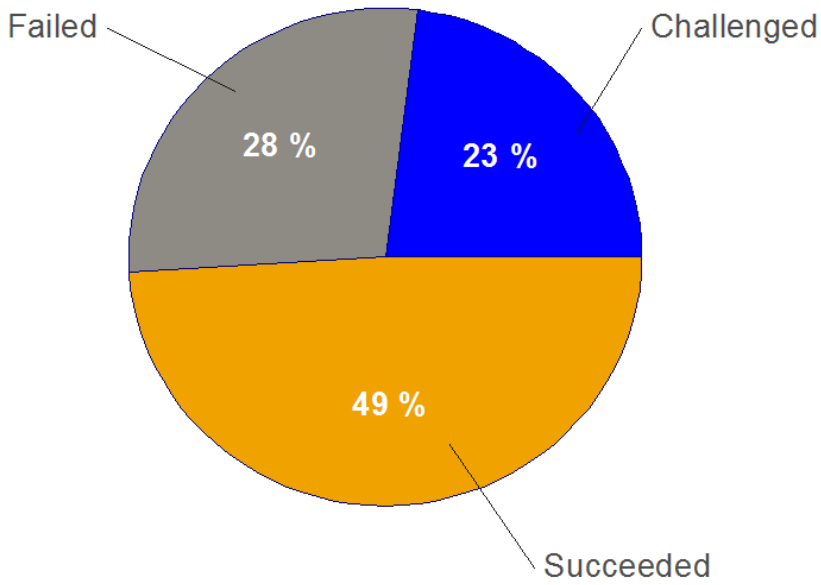
Risk Based Software Development – Reducing Risk and Increasing the Probability of Project Success

IT Software Development Just Isn't Working!

IT systems are at the heart of modern business and the development of new software applications and maintenance of existing systems are critical to productivity and profitability. Advances in software technology over the last 20 years have allowed progressively more complex business solutions to be created enabling companies to offer their customers exciting new services and products. And yet, software development projects still suffer from similar problems and characteristics, regardless of the technologies being used, that they suffered from more than ten years ago.

A report from the Standish Group clearly shows that the reliability of delivery of IT software applications has not improved over the years. Figure 1 is for 2001 and this looks remarkably similar to a diagram from the same source in the early 1990's.



Source: The Standish Group

Figure 1 – Software Projects

The important question that needs to be addressed is *why is software development so risky?*

Change Is Inevitable

All software projects implicitly have associated risks. One of the major sources of risk results from changes that occur during the project's lifecycle. In its most common form, this is seen as changing user requirements. It is, however, not just confined to this area. For example the following changes all present real risk to projects;

- Changes to the makeup of the project team or in stakeholders
- Changes in the technology being used
- Changes to any external systems with which the new software must work

How you deal with changes to the project is the key to reducing your development risks, and to increasing the overall chances of success of your project.

But what is the best way to do this? The major influence lies in the development process you choose and not the technologies being used.

Picking The Right Process

As a project manager you may be tempted to try and eliminate change in your project through a rigid policy of *no change*.

This is the case for waterfall processes where the major assumption is that development can proceed in an orderly, linear and predictable fashion. Waterfall processes have a clearly defined sequence of stages, each of which must be completed and signed off before progressing to the next stage.

For example Requirements (capturing) is completed before Analysis is started. Design cannot then be started until Analysis is complete and signed off…….and so on.

Figure 2 shows the typical lifecycle of a waterfall development process.

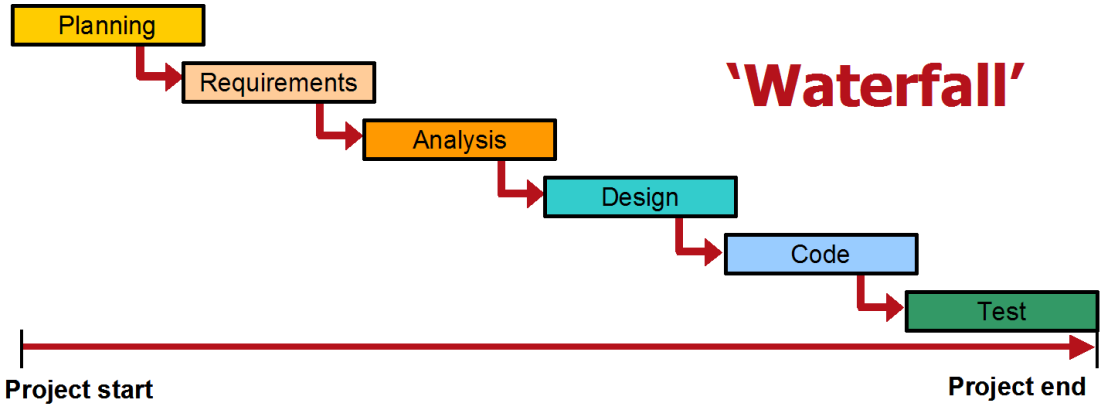


Figure 2 – Waterfall Development Lifecycle

Waterfall processes may appear at first glance to be a reasonable and common sense approach. The problem is that developing software is inherently unpredictable and this type of process does not cope effectively with change. The result is that this approach to developing software almost always fails to mitigate risk.

Worse still, with this type of process, you may all too often be lulled into to a false sense of security as the project progresses. By adopting a rigid sign off strategy for each activity in the project, it is all too easy to believe that the project is getting easier and risk is reducing as you get nearer to the go live date.

Change will inevitably occur during the project lifecycle, and this is usually the result of feedback. As we see in figure 3, waterfall processes have a late feedback cycle for the part that really counts, the actual software code, and not the signed off paperwork.

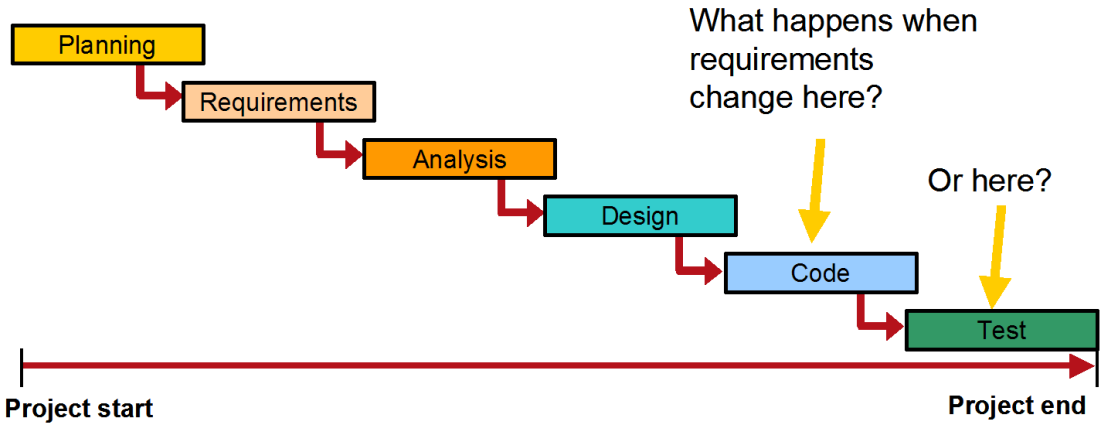


Figure 3 – Coping with Change

The result is that changes often need to be made to the system during the later project stages such as integration (when the different modules of software code are assembled into one application system) and the testing of the system proper. Imagine having to write off a 100 man-year project, after having already spent 95 man-years of the budget, because of a fundamental misunderstanding in the user's requirements!

The risk profile for waterfall processes, as shown in figure 4, is inappropriate for software development.

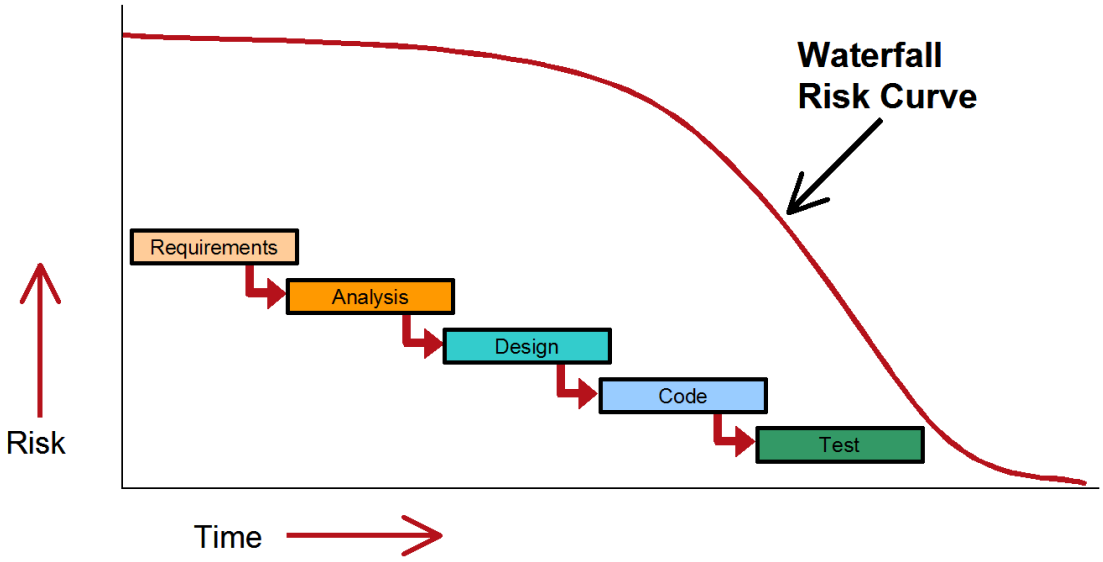


Figure 4 – Waterfall Risk Profile

The fundamental problem with waterfall processes is that they rely on the assumption that the progress of a software project can be predicted with reasonable accuracy from the outset, and that a reliable completion date can be derived from this. By assuming that the development lifecycle will be predictable, you are admitting that a high degree of risk cannot exist within a project. This is a fools paradise! The Waterfall process is the wrong approach for software development.

A Lower Risk Approach – Iterative Development

Project risks are associated with the unknown. These can be technical, organisational or business oriented in nature, such as:

- How will my system communicate with system A?
- How do I manage my offshore development group?

- Do the system requirements actually reflect the true needs of the users?
- etc….

Adaptive software development processes realise that producing software is a risky business and highly unpredictable in nature. They also recognise that the best mechanism for achieving successful delivery is to contain and mitigate risk by means of the regular and controlled delivery of testable software code to provide a reality check on progress and to give feedback to the users throughout the entire lifecycle of the project.

Key Principles

There is no single overriding rule that guarantees project success but the following key principles will greatly increase your chances:

- The adoption of an Iterative Risk Driven Approach
- The commitment and involvement of senior management
- A high level of communication between all project team members
- A highly skilled development team
- The adoption of a Use Case driven approach
- The adoption of a comprehensive and rigorous system of change control
- The use of Visual modelling

Iterative Risk Driven Approach

This will reduce risks by developing small pieces of the system over a short timeframe (a maximum of four weeks for large projects). At the end of each development cycle, the software should be demonstrated to the users to obtain feedback on its functionality and suitability.

An iterative approach will continuously reduce project risk from the outset. The reason why, is that it forces the team to address the most important aspects of functionality and to resolve high risk issues at an early stage.

A real example of how an iterative approach reduces risk occurred in 2001 when I was working on developing an application to monitor financial transactions. One of the users' requirements meant that the system had to process one million financial trades within an eight hour window. The client was adamant that this was a key requirement and that the application must be capable of running on low cost hardware. We recognised that there was a major risk that we might not be able to meet these performance objectives, so we built a basic working system within three weeks and provided critical feedback to the users that we could only achieve 10% of the performance required.

Result - the users reassessed their objectives and realised that a lower processing target was indeed adequate. This major risk was thus mitigated within the first three weeks of the project.

Figure 5 provides a comparison of typical risk profiles between iterative and waterfall development lifecycles. The iterative cycle continuously mitigates risk from the early project stages as a result of regular feedback from users.

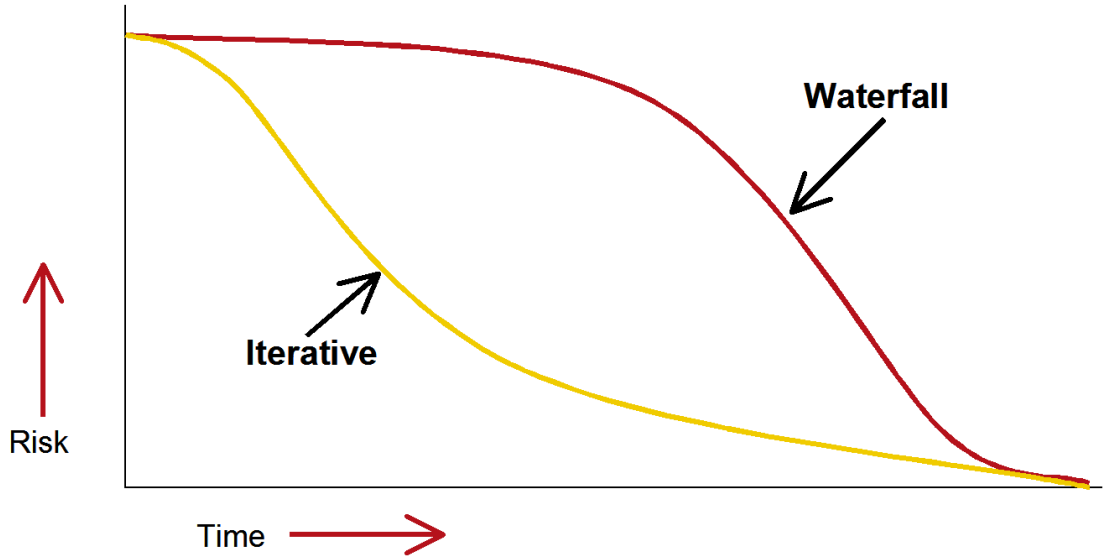


Figure 5 – Risk Profile comparison between iterative and waterfall processes

Senior Management Commitment

Changing the way people work is probably the hardest thing to do in any business and when it involves technologists it can be an even harder challenge. It is important to ensure you have full executive support for any type of process change. You will need backup to drive through changes in the early stages - but once the process starts working, people will naturally want to be associated with it.

High Level of Communication – Keep Talking!

First and foremost projects are about people. Without everybody working together you haven't got a team and the project will fail.

Communication is paramount within the team and more importantly with your users and sponsors. This has to be on a regular basis, daily with the team, and at least weekly with the users and sponsors. Your end goal should be to make all these groups part of the project team. Iterative development will greatly increase communication with the users. Communication comes in many forms - not just verbal, and includes producing good documentation. Aim to keep documentation lightweight by adopting a 'just enough' policy. Use visual modelling (key principle) whenever possible to raise the level of communication.

Highly Skilled Development Team

A highly skilled group of individuals that understands how to work together should form the nucleus of your team. They will be very efficient and productive. If you haven't already got the right mix of skills (technical and process), then invest in training focused on meeting your project's needs. Do not, however, waste time and money with off-the-shelf 'one size fits all' training courses. Consider supporting your project team with expert mentors to assist and accelerate process adoption.

Use Case Driven Approach

Use Cases are highly effective because they provide a contextual representation of the system requirements. One of the main reasons why they should be used is that they are non-technical in nature and provide a written step by step description of how the actors (system users – human or otherwise) shall interact with the system. Simplicity is the key strength of use cases.

So many times I have seen documents that purport to contain the complete system requirements, when in fact they are just a set of business rules and a list of user needs. To

drive a project forward, you need objectives and goals to aim for, and Use Cases are the best way do this. They bind together the whole development process from the capturing of user requirements through to the testing of the application software.

Change Control

This is very different from stopping change from occurring to your project. Instead the emphasis is on the careful consideration of new requests and deciding (with the users) how these should be accommodated. For example, if the delivery date is fixed, does a new request mean that an existing requirement has to be removed from the live release, or that a further system release should be considered?

Visual Modelling - 'A picture is worth a thousand words'

Visual modelling is the modern equivalent of the "flowcharts" used in the early days of software production. The value of visual aids remains valid and diagrams should be used throughout the project. Demand that all roles use the same modelling notation, from business analysts to designer and coders through to testers. The UML (Unified Modelling Language) is the de facto standard for the visual modelling of systems and should be used to communicate consistently and concisely across the project. There is no excuse for not using it.

Are You Really Risk Focused?

I have worked with many clients and spoken to senior managers and technical specialists who swear that they are risk focused and follow an iterative approach. You will come across the same situations, so how do you know if your IT group is really doing risk based iterative development?

Ask a few telling questions:

How long is an iteration?

The answer should be anywhere from a few days to a maximum of about four weeks. Answers of three months or more should indicate the process is still waterfall with long feedback cycles with the associated high risk of getting it badly wrong before being able to rectify the situation.

What is delivered at the end of each iteration?

The answer should be a demonstration of some actual working software to the users, during which they will have the opportunity to provide feedback. If the answer is documents for sign-off, then they are missing the point.

When do you the users see what they are going to get?

The answer should be at the end of every (short) iteration. An answer of "only during acceptance testing prior to delivery" is just not acceptable.

When do you intend to start integration testing?

The answer should be that integration testing is a continuous activity that occurs during each iteration.

How do you manage risk?
The answer should be that there is a regular assessment of risk, quantified in terms of probability and impact to the development schedule along with proposed mitigation strategies. The project manager should maintain a running risk list, available for inspection by all people associated with the project.

Longer Term Business Continuity

So if I adopt all these good practices what does this really mean to my business?

We know that software development is a highly risky business that costs a lot of time, effort and money. But the greatest costs occur long after the initial development is over, as shown in diagram 6, when the system enters the support and maintenance phase of it lifecycle.

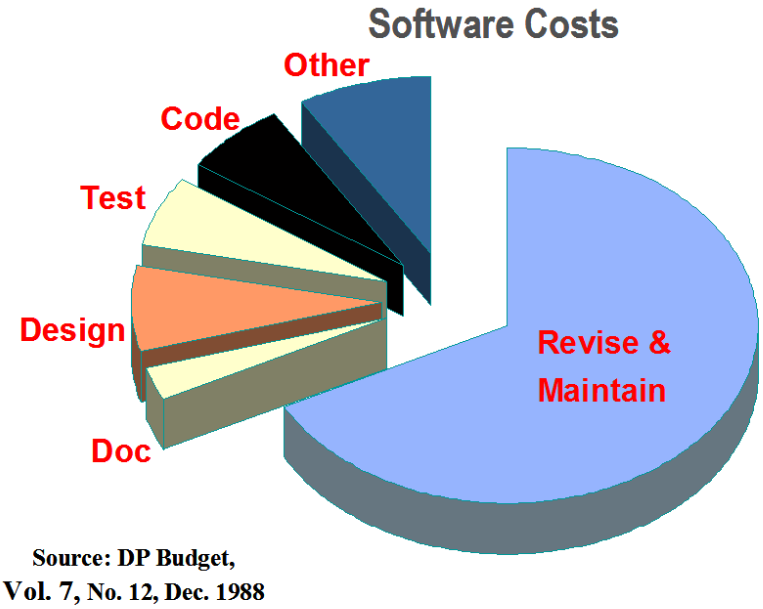


Diagram 6 – Software cost breakdown

Adopting these good practices, with the focus on developing systems that deliver what the users actually want, will usually produce systems that are of a higher build quality and are 'fit for purpose'.
This typically results in systems that are easier to maintain and revise and thereby reduce the long term operational costs and associated risks.

Cliff Murphy is a founder and director of Liemur Limited.
Liemur is a UK based company providing a range of specialised services to optimise software development and maximise business return on investment in IT.

Email: cliff.murphy@liemur.com
Web: www.liemur.com

Tel: 0870 620 8131
International +44 1243 372999

Address: Liemur Limited
74 Shrewsbury Lane
London
SE18 3JL