Chapter 10. Project Planning

Table of Contents

Objectives

At the end of this chapter you will have acquired an introductory understanding of what software and software engineering are, and some of the common myths surrounding software engineering.

Why do systems fail?

Information systems are generally classed as failures if they are either unsuccessful as a product and/or unsuccessful in their production. For instance, a system can be a poor product because of:

- Failure to meet requirements
- Poor performance
- Poor reliability
- Poor usability (the extent to which a system makes it possible, and easy, for a user to perform a task is a complete field of computing in its own right, since so many systems fail to do this well)

However, it may be a failure in production because it was not produced on time and to budget. It can sometimes be the case that because a system was a failure in production, corners were cut during development in an effort to minimise cost and time overruns but as a result the product was not properly developed and turned out to be a failure as well.

There can be no hard and fast rule which will say that a system is a failure, although a system that is 100% over budget, or past its deadline could well be considered a candidate for being defined as a failed, or failing system.

It seems remarkable that any systems should be failures. Surely, in planning a project, sufficient attention is paid to the above to ensure a successful system. After all, computing is meant to be a science. How then is it possible that there are some spectacularly unsuccessful systems?

Very little concrete research has been done on why systems are badly produced. This is not too surprising - businesses do not want to put themselves up as examples of failure. Much evidence is therefore anecdotal though there are an increasing number of case studies available. More wide-reaching research was done by KPMG who sent out questionnaires on software runaways in 1989 and 1995.

System Failure Factors

Fortunately, there is a lot of agreement between the research and less formal experience. The main factors that are behind a system failing are:

- Shifting requirements

- Bad estimation
- Bad management
- New technology

Each of these factors shall be discussed individually, however, they all could also be described via the single issue of complexity. Computers are a universal tool — theoretically they can perform any algorithm of which we can conceive and as time passes the practical limitations are pushed further and further back. It is this almost unbounded flexibility which is what makes computers useful. They have the capacity to manage complexity and bring staggeringly huge or enormously difficult processes under our control. Handling complexity is an essential feature of modern software.

Shifting requirements

When a client first describes the requirements for their new information system, they really only address the issues that they are aware of at the time. The process of developing the new system may bring new issues to light either through trying to formalise requirements or through seeing early versions of the system. As a result, the client may change their requirements mid-way through development. Because a computer is such a flexible tool, this is not perceived as unreasonable. But a change in requirements can mean that the code that exists is no longer suitable and, in effect, some or all of the development must start again.

Bad estimation

Estimating the schedule and budget of a large software project is difficult. Often projects are designed to do what has never been done before. Even when some projects are based on others, there is usually some change in environment or requirement which the existing system cannot address. As a result of this inherent novelty, there are no hard and fast rules for working out how long a project will take and how much it will cost. Some people work on the expected number of lines of code, others on the number of requirements, others on the number of modules. All of these are helpful, none are 100% accurate and there are always examples where they are down right wrong.

The difficulty in estimation is compounded by software development being a relatively new human endeavour. Other construction and engineering tasks have built up over a period of centuries. Software engineering is only forty years old and in that time the technology has changed so rapidly that is difficult for any stable base of experience to have been accumulated. Often the best estimations can only be produced by people who have been involved in software development for a long time.

New technology

Possibly the most surprising cause of failure is new technology. New technology may come in the form of software, such as new programming languages, or network communication techniques, or hardware such as more powerful kinds of computer or storage devices. It adds prestige to say that a project is using the latest technology and so businesses are often keen to incorporate it into their latest software development in one form or another. But the technology is not always appropriate. No matter how new and exciting a technology may be, if it is not right for the job then it will only cause problems. If its not yet fully developed then there may be no support for any problems which arise and it may even be fundamentally incapable of the task.

Technology is also a source of complexity in itself and it is advancing at an enormous rate. Moore's Law states that there is a ten-fold increase in computing power every 18 months. It is changing far too rapidly for any one person to stay ahead of all of the developments for any length of time. So using new technology often means needing new expertise through training or hiring new staff. Both of these can cause increases in cost and schedule of a project.

Bad management

Experience is also key to good management of software projects. The complexity of software projects involving hundreds of thousands or even millions of lines of code is far more than any one person is

able to grasp or work with effectively. For large projects managers divide staff into teams of developers to break down the system development into smaller, self-contained tasks which each team, and within it each developer tackles.

The project manager must ensure that the activities of the team are co-ordinated and that everybody is proceeding towards the required goals. First and foremost this requires a good communication network between developers and the larger the team the more communication is required. If this breaks down, the project is almost certainly doomed.

A manager must also cope with external pressures such as changing requirements, technological problems or even mismanagement elsewhere. If all suggested changes were to be taken on board, development would never progress but constantly get stuck trying to incorporate the latest change. But at the same time, neglecting to include important changes could result in a flawed system. Ultimately, experience is the only resource which can help a manger handle these difficulties.

As with anything, it is not usually the case that any one of these factors alone caused the failure of a software development. And the factors interact and can even exacerbate each other. But at the end of the day, it is the enormous complexity of software projects which is at the root of systems failing.

The issue of complexity is discussed well in the opening chapter of most books on object oriented software analysis and design.

Review

Questions

Review Question 1

What are some examples of why software systems are considered a failure? And the factors that can cause such a failure?

A discussion of this question can be found at the end of the chapter.

Review Question 2

For each of problem you identified in the previous question, suggest how it could be avoided or overcome.

A discussion of this question can be found at the end of the chapter.

Answers

Discussion of Review Question 1

Software might be considered a failure if it:

- fails to meet its requirements,
- performs poorly,
- shows poor reliability,
- has poor usability,
- is not produced on time or within budget.

These failures can be caused by a number of factors:

- Shifting requirements
- Bad estimation
- Bad management
- New technology

Discussion of Review Question 9

There are no straightforward answers to these questions. The whole of this module is about how the computing professional has investigated software systems and their development, and is working towards more rigours and engineering approaches to avoid or be able to address the problems that have been identified.

Here we shall make some informal suggestions about how each problem might be avoided and addressed. You might wish to revisit this question as you work through the module, to see how your understanding of system analysis and design informs your view on the problems and their solutions.

- Shifting requirements
  - Getting the requirements at the beginning of the project is vital, which must involve close work with the system sponsor and users
  - Being able to represent the requirements in some rigorous but non-technical way (so users can understand and correct any errors) is important
  - The system sponsors need to be made to understand the costs if requirements change during the project (more work, more time etc.)
  - The consequences to the development of a new system must be thoroughly investigated and the system re-designed if any changes are made to requirements — this may even mean abandoning completed work and redeveloping models and designs from scratch
- Bad estimation
  - As the profession develops more experience, and new 'metrics' for system development and costs, estimate should improve
  - It must be remembered that system analysis and design is not a formal science, nor can it be, since it involves communication with humans, and the learning of how organisations current work and might work better in the future
- Bad management
  - The use of project staff with decision making authority who have experience of similar, successful projects is a desirable position when choosing project staff
  - Measures need to be in place (such as a set of 'milestone' documents and delivered system components) to give the sponsors and project managers feedback on project progress
  - If problems can be identified early they can be taken into account and the effects minimised
- New technology
  - Careful choices need to be made, considering the potential benefits of new technology with the costs and a dangers of skilled staff lack of support or knowledge of new problems
  - As a general rule new technology should only be used where there is a significant perceived benefit and where expertise is available for the duration of the project