

The Requirements Management Challenge

Dr. Ian Faulconbridge
Magpie Applied Technology © 2009
admin@magpietech.com.au

Abstract: Without an adequate requirements management process, projects are left with poorly defined scope, sketchy estimates of cost and schedule. Frustrations occur due to a seemingly constant flow of changes to the project scope throughout the project. When the project does finally end, it is difficult to confirm that a quality solution has been delivered. Then come the problems associated with supporting the system throughout its operational life. This White Paper discusses the concept of project requirement management. The structure provides content in the left hand column and key points (*italicised*) in the right hand column.

Introduction

The important role of requirements and requirements management was recognised in some of the large United States Department of Defense (DoD) projects that emerged after WWII. These were projects that had some of the hallmarks of difficult projects that we still struggle to manage today.

These projects usually involved complex and challenging stakeholder requirements and expectations. Even though the stakeholders had high expectations for the outcomes of these projects, the stakeholders were not able to articulate their requirements for the capability completely and unambiguously.

...high expectations

*...incomplete and
ambiguous
requirements*

During the 1950s and 1960s, exciting, emerging technology was being discussed. Stakeholders were eager to take advantage of the enhanced levels of function and performance that these emerging technologies promised. This may have exacerbated the problems associated with stakeholder requirements, as the stakeholders tended to request levels of function and performance that they anticipated the new technology could deliver, rather than concentrating on the function and performance that they needed the capability to deliver.

*...unable to
concentrate on the
function and
performance that they
really needed...*

With increasing project complexity came the need for larger numbers of technical disciplines and specialists. Each discipline and specialist group tended to bring to the project their own lexicons and embedded processes, often causing clashes, arguments, ambiguities and misunderstandings. The problems associated with integrating technical disciplines often resulted in problems that were incorrectly called “technical problems”.

*...large number of
technical disciplines
and specialists...
...clashes, arguments,
ambiguities and
misunderstandings.*

Owing to some of these issues and others, these projects were considered to be “risky”.

...risky...

The more successful of these projects started to come up with ways of managing some of these issues. Appreciating the importance of requirements and the role played by requirements in a project environment was one of the outcomes of this period of time. Additionally, understanding how requirements are developed throughout the project lifecycle and how those requirements influence critical project management issues such as scope, cost and quality also started to be more appreciated by project management specialists.

... understanding how requirements are developed throughout the project lifecycle...

...how those requirements influence critical project management issues such as scope, cost and quality...

Eventually, the more successful processes started to be written down and standardised to allow the more successful processes to be repeated on different projects. Lessons learnt via additional projects were fed back into the formalised processes in order to continuously improve the results. One of the earliest forms of these processes is in the form of a US DoD military standard called MIL-STD-499 (1969).

...continuously improving, formalised processes...

The US DoD called this process *systems engineering*; a term that survives today to describe the rigorous, robust and repeatable process that documents desired capability requirements and methodically translates those requirements into a solution.

Systems Engineering Explained

The term “systems engineering” often disengages people who are not from a technical or engineering background. The term is often associated with military systems such as aircraft, tanks and warships. There is a tendency in some circles to associate systems engineering with needless and costly process and complexity. Whatever your background, industry or current attitude towards the term, there is no need to be intimidated or disengaged by the words “systems engineering”. There is value in exploring the two constituent terms though:

...systems engineering helps bring about solutions to complex problems.

- A *system* is a complex set of connected things or parts arranged to work together to form a functional whole. It may be easier to remember a *system* as being a solution to a complex problem.
- *Engineering*, when used as a verb rather than to describe a profession, means to contrive, arrange or to bring about.

In short, then, systems engineering is all about bringing about a solution to a complex problem.

Solving Complex Problems – More Than Equipment

Historically, as mentioned earlier, people have often associated systems engineering processes with producing some piece of military hardware. This is drastically under-selling what systems engineering can do for you during complex procurements or capability development exercises, and it is also dangerously under-scoping what constitutes a solution to a complex problem.

...associating systems engineering with military hardware drastically undersells what it can do for you.

In order to bring about successful solutions to our complex problems, we need first to be able to understand the need or problem that we are trying to solve. Based on that need, we are able to use processes to extract a comprehensive understanding of exactly what the solution to our problem needs to be able to do; how well it needs to perform; under what conditions it needs to perform; and what other systems are involved in its operation. This thorough understanding of our requirements puts us in a position to consider the solution options that are available to us, before selecting the preferred solution. The preferred solution can then be specified, designed and developed, tested and brought into service, thus satisfying our need. Following acceptance into service, the solution needs to be supported and maintained until it no longer meets our needs. This is what systems engineering (or solving complex technical problems) is all about.

...understand your need (or problem) before selecting the preferred solution.

What about the concept of a solution? Does a piece of equipment solve a complex problem? Invariably the answer is no. Equipment needs to be operated and supported by trained people. We also need to recognise that people come and people go from our organisation which results in the ongoing need to provide training and development to ensure that there is a continuous supply of people who are qualified to maintain, support and operate the system. Our organisation also needs to develop and maintain procedures and rules regarding how the system is used. These rules guide the operators and also provide the basis for measuring whether the system is meeting our requirements. There is clearly more to a complex system than just a piece of hardware or software.

...a solution to a complex problem invariably requires more than a new piece of equipment.

A useful memory-jogger for this concept is to remember POSTED:

- **P**ersonnel
- **O**rganisation
- **S**upport
- **T**raining
- **E**quipment
- **D**octrine

...think POSTED when considering complete solutions.

Requirements and the Solution Lifecycle

The nature, level of detail, and role played by requirements change as the project moves through its logical progression. The logical progression of a project is usually measured against a lifecycle of some sort. One of the key benefits of selecting and using a suitable lifecycle model is that the changing role of requirements can be acknowledged and managed.

..having a defined lifecycle model aids requirements management.

An additional problem with some lifecycle models used in project management is that they focus on the acquisition stage of the system lifecycle, sometimes at the expense of phases that should occur prior to acquisition and phases that will follow acquisition. Establishing the need for a project and clearly understanding the constraints and requirements must occur prior to acquisition. Similarly, the requirements associated with operation, support and disposal of a system should also be appreciated prior to commencing an acquisition. In this way, the acquisition is conducted with operation and support in mind.

..critical phases must occur BEFORE establishing a project.

..critical phases that occur FOLLOWING the project must be considered early

Different industries tend to adopt different lifecycle models for their particular projects. There are a number of useful references^{1 2 3 4} that offer guidance when deciding upon the best lifecycle model to use in different circumstances. These references are standards-based, but other models will exist in company policy and procedure, published books, journal and conference papers, and industry best practice guides.

..many reference sources exist to help define a lifecycle model.

Becoming familiar with a specific lifecycle model, understanding what happens in each phase, and appreciating the changing role of project requirements as a function of phase is a useful exercise. Once familiar with one model, it is usually easy to adapt to other lifecycle models as required. A useful starting point is illustrated in Figure 1.

..being familiar with one lifecycle model makes adapting to other models easy.

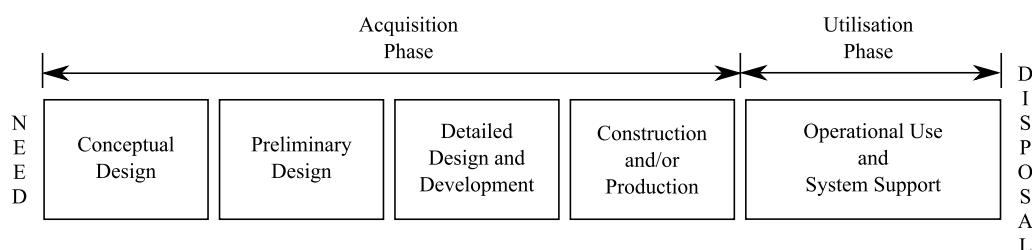


Figure 1. A Simple Lifecycle Model⁵

¹ PMBOK, *A Guide to the Project Management Body of Knowledge*, Upper Darby, Project Management Institute, 1996.

² IEEE-STD-1220-2005, *IEEE Standard for Application and Management of the Systems Engineering Process*, New York, IEEE Computer Society, 2005.

³ ANSI/EIA-632-1998, *Processes for Engineering a System*, Washington D.C., Electronic Industries Alliance, 1999.

⁴ ISO/IEC 15288:2008, *Systems and software engineering -- System life cycle processes*, International Organization for Standardization, 2008.

⁵ Faulconbridge and Ryan, *Engineering a System: Managing Complex Technical Projects*, Canberra, Argos Press, 2005.

The Underlying Process

Establishing requirements and making the necessary technical decisions throughout the lifecycle in order to solve a complex problem tends to make ongoing use of a very simple problem-solving process, as illustrated in Figure 2.

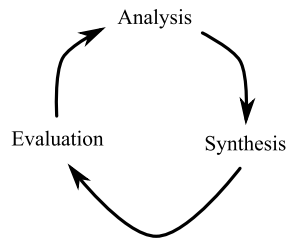


Figure 2. Simple Problem-solving process⁶

The term *analysis* refers to determining information about the particular problem. In the early stages of the lifecycle, this is focussed on trying to establish the need for a project and articulating the functional requirements that the eventual solution will need to meet, in order to fulfil the need. *Synthesis* is a term used to describe the evolving solution space. In the early stages, synthesis will be used to determine possible solutions to the broad problem, but in the later stages of projects, synthesis may involve very detailed design outputs like software code and hardware designs. *Evaluation* essentially closes the problem-solving loop, by assessing that the solution options delivered by the synthesis process have addressed the problem space as determined by the analysis activities. Often, we will have to complete the *analysis-synthesis-evaluation (ASE)* loop a number of times before we are satisfied.

It is critical to note that this generic process is that it is applied iteratively throughout the lifecycle model; each application building on the previous and each application having its own objective. Typically, the customer applies the ASE loop during conceptual design to establish the need and requirements from the customer's perspective, and to explore possible solutions. During subsequent phases, a contractor re-applies a similar process (a number of times) to establish more detailed (derived) requirements; to refine their proposed solution; and to confirm that their design is adequate. Whilst the underlying process remains unchanged, the reason for applying it; the output of its application; and the contractual entity responsible for its application changes.

*...use a very simple
problem-solving
process*

*...analysis provides
information about the
problem.*

*...synthesis is evolving
design of the solution.*

*...evaluation confirms
the adequacy of the
solution.*

*...the simple process
is applied (iteratively)
throughout the
lifecycle.*

⁶ Faulconbridge and Ryan, *Engineering a System: Managing Complex Technical Projects*, Canberra, Argos Press, 2005.

The concept of the iterative application of the process across the lifecycle is illustrated in Figure 3.

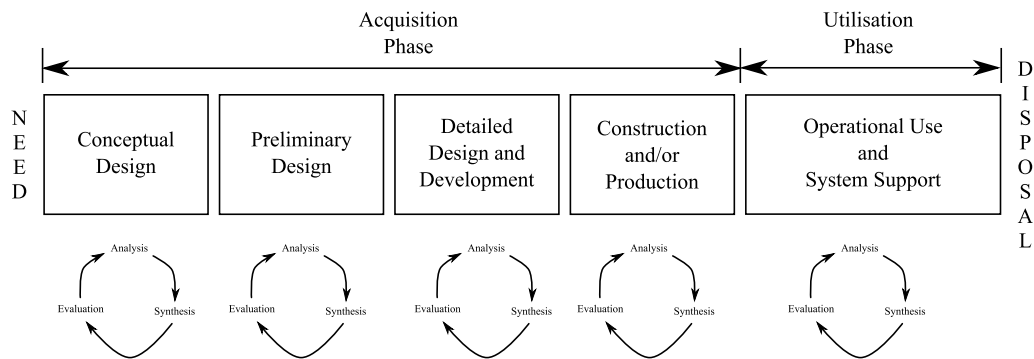


Figure 3. Iterative application of the ASE loop

Rationale – Why Worry About Project Requirements?

There are some obvious answers to this question and maybe you should confirm your understanding of why we bother spending so much effort on understanding and managing our requirements. Apart from the obvious answers that you will come up with, there is a more compelling reason why we need to work hard on our requirements management right from the very early stages of our projects. The fact is that fixing mistakes gets increasingly difficult and expensive as we proceed through the lifecycle. At the early stages (during conceptual design), changing our requirements by adding, modifying or deleting requirements can be done relatively easily and cheaply. It might only cost someone's time to make the necessary changes and seek stakeholder endorsement. Once the requirements become embedded in a contract, the ability to make changes starts to become impaired. Early in the contract (during preliminary design), it will generally be possible to make changes to our requirements via some form of contract change process. The contractor will generally charge the customer some money and seek some schedule relief for this change, as the contractor may have to re-work aspects of their design and documentation. Later in the design process, making changes becomes much more difficult, expensive and time-consuming. By the time the design is being realised through the production and construction process, making changes to our requirements becomes almost impossible. When the system is finally transitioned into the utilisation phase, changes may not be possible.

..fixing mistakes gets increasingly difficult and expensive as we proceed.

In quite a well-known study, Boehm⁷ investigated the relative cost of rectifying mistakes in software-intensive projects as a function of lifecycle phase. His findings are summarised in Figure 4 and show that getting our requirements right as early as possible pays off in the long run. The cost of fixing mistakes early in the lifecycle does not compare to fixing them after the system has fielded.

...statistics confirm the importance of requirements

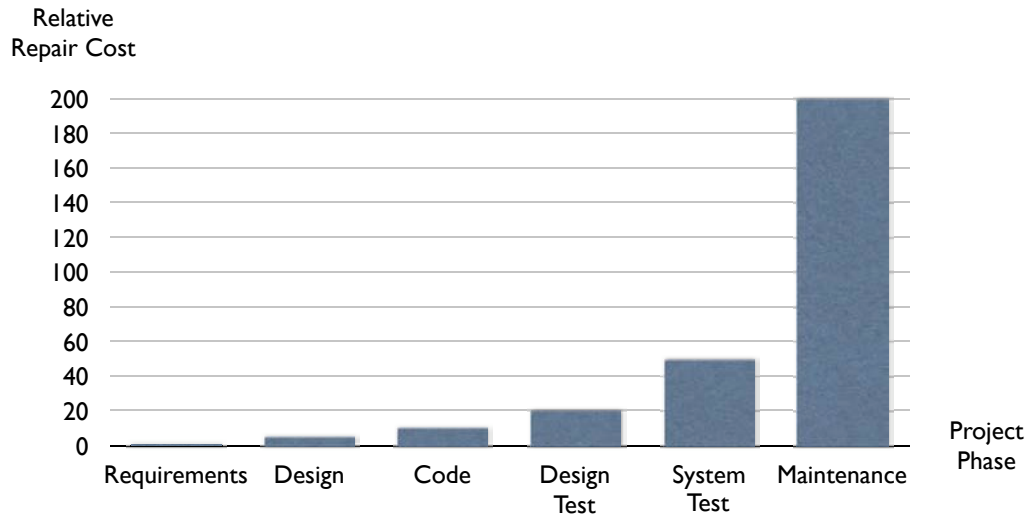


Figure 4. Relative cost to repair software errors

Although dated, the study has been replicated (and therefore validated) many times since its initial publication. The concept and message applies to projects regardless of whether they are technical projects involving significant engineering development or not. We simply must get our requirements sorted out early and continue to manage them actively throughout the lifecycle. Our projects' success depends on it.

...get our requirements sorted out early and continue to manage them...

About Magpie Applied Technology

Magpie is a consultancy that specialises in applying systems thinking to capability development projects across a range of industries. Magpie also offers mentoring and training courses in systems engineering and requirements management. Magpie's consultants are professional engineers who combine strong theoretical knowledge across a variety of technical disciplines with practical, real-world experience in order to deliver pragmatic solutions.

For more information, visit: magpietech.com.au

⁷ Boehm, B., *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.