White Paper : A Primer on Requirements Engineering
Introducing Beavers Requirements Engineering Services Capability

Beaver Computer Consultants staff have over 10 years of requirements engineering experience, predominately involving systems that were mission or safety critical, and where mature requirements management is a necessary part of the process, assessed by acceptance authorities and regulatory bodies in order to allow a system to come into operation. We have developed specifications, undertaken verification and validation activities, defined processes and customised tools to implement best practise requirements engineering, and provided training and mentoring. Our experience can be used to develop business and control systems. Our approach addresses many topics raised in recent reports on avoiding project failures.

Beaver Computer Consultants Ltd
11 Button Road
Grays
Essex
RM17 5HE

Phone : +44 (0) 1375 377235
Fax : +44 (0) 1375 377235

Email : paul@beaver-consulting.co.uk
www.beaver-consulting.co.uk

home of

www.managing-requirements.com
www.project-support-office.com

Limitations of Use and Copyright Notice

Change History

Author	**Comments**	**Checked**	**Status**	**Version**
P Ransley	Document Creation		For Information Only – No Formal Release	0.1
P Ransley	Contact info changed (emails and web addresses). Section 10 (Change Control) has been expanded. Information Model Diagram added in Section		For Information Only – No Formal Release	0.2
P Ransley	Minor wording changes and hyperlink updates.		For Information Only – No Formal Release. September 2003.	0.3
P Ransley	Added draft process models for subsystem processes, layers of requirements specification, and use of modelling to derive, decompose and verify requirements.		For Information Only – No Formal Release. September 2003.	0.4

Change Prognosis

Reason	**Status**
Section 4.8 diagrams will be added to table to show evolution of the information model.	Open
Major revision of grammar. Move domain concepts section to an appendix. Explain differences between requirements documents and machine specifications.	Open

Contents

List of Tables

List of Figures

1. Summary

This white paper starts by presenting a summary of research findings on critical success factors derived from projects that deliver the business need, on time, and to budget. These factors are contrasted to projects that fail. This allows us to establish why requirements engineering is important.

To understand requirements engineering we first review the definition of what a requirement is, the categories of requirements, and the types of information or attributes that are associated with requirements. Requirements are written into a specification. The boundary of what the specification represents is discussed with reference to the importance of understanding the domain application or environment in which the solution will need to work. The types and contents of various specifications are out lined, together with a summary of the tasks necessary to populate a specification. We introduce some key concepts including;

- the types of specifications and the relationship between the specifications, particularly with respect to compliance, and the role of traceability
- approaches to the structuring of specifications and documents to create an underlying information model
- agreeing the contents of each specification and the structuring of information in it
- the processes necessary to develop requirements, test them, under take compliance monitoring activities, and manage changes
- maximising the use of information captured such as in progress reporting, measuring the quality of requirements, and supporting change management.

Finally we present a process model, starting with the elicitation of the initial stakeholder need, to writing and testing decomposed requirements, and then monitoring compliance and acceptance as the solution is developed and delivered. Each process step is illustrated with sample attributes and factors that need to be considered. We provide references to books and internet resources and pose questions with suggested answers or remedy's as to how to handle particular issues based on real world requirements engineering experience.

2. What is special about requirements?

"When I use a word," Humpty Dumpty said in rather a scornful tone. "It means just what I choose it to mean - neither more or less."
"The question is," said Alice, "whether you can make words mean so many different things."
"The question is," said Humpty Dumpty, "which is to be master - that's all."

Charles Lutwidge Dodgson (Lewis Carroll) (1832-1898)

A user needs a new solution, comprising of software, hardware, procedures and perhaps other equipment as well, and work with people and interface to external systems and processes. The designer and supplier aim to meet this users need. The method of communication is by using a contracts document or a specification. Here lies the problem, the words in the document can mean different things to different people, or the wording, or rather its meaning, can be manipulated by each party to suit there own circumstances.

What problems can arise? Most often the user, business client, procurement manager, and supplier are not talking about the same thing, each misunderstand each other, but believe they are consistent. Even when they do understand each others intentions, in practice the solution does not match the need, or deliver the assumed benefits because key factors were not understood or were not identified to start with. An end user who is not satisfied with the solution can use a poorly worded requirement or one that was not realistic, needed, or testable as the basis for not accepting the solution.

Such problems will add time and cost to the development process as solutions are reworked and issues debated and often not resolved until late in the day.

Requirements engineering as a discipline seeks to minimise these types of problems by using systematic and structured processes working from an early stage in the development life cycle. It gives us a better understanding of what must be delivered through good structuring and detailing of the relationships between requirements and supporting information, measuring the realisation of the solution, and driving the acceptance process. The overall aim is to ensure the smooth flow of information between users, designers, and installers so minimising risk, providing early confidence that the solution will work, and that it addresses the relevant need or problem,.

This paper is a primer on the discipline of requirements engineering covering their elicitation, analysis and management through the life cycle. An approach, based on a more detailed process as used by Beaver Computer Consultants, is outlined.

3. The role of requirements in project failures and successes

First though it would be useful to understand why requirements are so important. Research from the Standish Group and others, shows that on average 80% of projects fail in some way. The findings are summarised in Table 2. Things are further complicated we look at the size of a software project we start to see how quickly project delays or cancellations occur as the number of function points increase as shown in Table 1. Larger projects have more requirements, with greater complexity, and require more compliance or verification or validation data to provide assurance the project is working on the right problem and that the solution delivers the need.

Table 1 - Failure as a function of project size

Function Point	Early	On Time	Delayed	Cancelled
1 FP	14.68%	83.16%	1.92%	0.25%
10 FP	11.08%	81.25%	5.67%	2%
100 FP	6.06%	74.77%	11.83%	7.33%
1000 FP	1.24%	60.76%	17.67%	20.33%
10,000 FP	0.14%	28.03%	23.83%	48%
100,000 FP	0%	13.67%	21.33%	65%
Average	5.53%	56.94%	13.71%	23.82%

Source : Capers Jones, Software Systems Failure & Success

While some of the research may be a few years out of date now, the message is still clear, a good process needs to focus on strong requirements management, complemented with user involvement, top down leadership, and realistic planning. The larger the project, the more complex are the issues of managing requirements, tests, and change.

Table 2 - Project Success & Failure Factors

Classic Project Success		**Project Challenged Factors**		**Project Impaired Factors**	
Factor	% Responses	Factor	% Responses	Factor	% Responses
User Involvement	15.9%	Lack of User Input	12.8%	Incomplete Requirements	13.1%
Executive Management Support	13.9%	Incomplete Requirements & Specifications	12.3%	Lack of User Involvement	12.4%
Clear Statement of Requirements	13.0%	Changing Requirements & Specifications	11.8%	Lack of Resources	10.6%
Proper Planning	9.6%	Lack of Executive Support	7.5%	Unrealistic Expectations	9.9%
Realistic Expectations	8.2%	Technology Incompetence	7.0%	Lack of Executive Support	9.3%
Smaller Project Milestones	7.7%	Lack of Resources	6.4%	Changing Requirements & Specifications	8.7%
Competent Staff	7.2%	Unrealistic Expectations	5.9%	Lack of Planning	8.1%
Ownership	5.3%	Unclear Objectives	5.3%	Didn't Need It Any Longer	7.5%
Clear Vision & Objectives	2.9%	Unrealistic Time Frames	4.3%	Lack of IT Management	6.2%
Hard-Working Focused Staff	2.4%	New Technology	3.7%	Technology Illiteracy	4.3%
Other	13.9%	Other	23.0%	Other	9.9%

Source : Standish Group (www.standish.com)

While this paper focuses on the requirements engineering aspects, requirements engineering works best when combined with an iterative, risk based development method using visual modelling, such as those now typified by the Rational Unified Process, Dynamic Structured Development Method, Artisan's Real Time Perspective, and PRINCE. Other white papers by Beaver Computer Consultants address business and systems analysis, including UML, and project management.

So we can see that a critical success factor in successful projects is the use of good requirements management principles, placed under change control, and integrated with other project disciplines. Success is even more likely when this occurs in an open environment with collaborative working between the business, the users, and the developers. In this context requirements management can be seen as a risk mitigation activity. It is worth remembering the other critical success factors include utilising specialist rather than generalist staff and thorough reviews of the technologies to be utilised, including an assessment of the maturity and development risk associated with them, representing a tight link between requirements and acceptance processes.

Requirements engineering as part of a defined development process is fundamental to when complying with standards such as ISO BS 9001 and software development capability maturity models, such as the Software Engineering Institutes Capability Maturity Model (SEI CMM) and the European SPICE model. A process is a prerequisite to repeatability, as it allows project experience to be fed back in by refining the process or peoples knowledge and skill in implementing it, and creates an auditable trail showing the identification of the problem to work on and the decisions made to derive and deliver the solution.

While some people believe they are addressing requirements engineering on their projects, our experience is the level of detail, the methods used, and the way in which the information is used within any control and decision making process does not constitute good practice. Requirements engineering does require additional effort but this is paid back with better understanding and delivery of solutions that correctly meet the needs of the business and users within the defined constraints. Its better to feel the pain from problems early on when something can be done than discovering them at the end of the project when they can not be dealt with.

The requirements concepts put forward here form an approach which is used by Beaver Computer Consultants in various modified forms. This paper will introduce you to a number of key concepts, terms, and methods to allow you to develop a requirements engineering process and allow us to demonstrate our capability to you.

4. Information and requirements types and terms

4.1 Definition of a requirement

What is a requirement? Many people will say its what's in a specification. However it is worth returning to first principles and examining several definitions. The IEEE Standard Glossary of Software Engineering Terminology (1997) defines a requirement as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

Sommerville and Sawyer (1997) add some important points "Requirements are…a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system".

What we can see then is that many people have ideas on what requirements are, but perhaps what's most important is that they are recorded, analysed, and agreed. Particularly with the stakeholders and users.

4.2 Types of Information and Requirements

On a project we can collect a lot of requirements and related information, even on a small project 200-300 is usual, on an average MIS solution 10,000 requirements and related pieces of information is not uncommon. What is more important to remember though is that not everything in the specification is a requirement. We need to see the trees in the wood. Beaver Computer Consultants break down any existing document text, specification, or even verbal statements into a number of categories which are:

Table 3 - Types of Requirements

Type	**Category**	**Description**
Requirement	Functional	What service needs to be achieved by the system
	Non-functional or performance	How well a function or set of functions needs to perform such as portability, timing, reliability, or throughput. Some standards include organisational, management, legal and similar requirements under this heading or separate out in other types.
	Constraint	A limitation that limits the problem space,

		such as a particular technology must or must be used, that certain architectures are not allowed (as in safety SIL's), related to functionality or domain knowledge. Notice already a link can exist between types of information.
	Process or management	The method of realising the solution, of monitoring progress, or of gaining acceptance of the system. Processes can include commercial terms, damages, defect periods, report periods, and the variation / waiver process to be used.
Domain Knowledge		A fact that is outside the system boundary, but must be true pr the solution must comply with the domain knowledge, for the solution to be successfully realised.
Definition		Important words and abbreviations are defined. Definitions can include project stages, acceptance criteria. Its important not to confuse these will a functional or process requirement.
Context or Headings		Text that gives background information, improves the flow of text or requirements, and makes the document readable. Often contextual information surrounds the requirement.

Where functional, non-functional, constrain and process requirements are defined secondary categorises can be applied, and used to allocate requirements to particularly work groups, understand what documents and processes are requirements, and manage the information in a more efficient manner. Secondary categories can include;

- core
- safety
- reliability, availability, maintainability
- security
- installabilty
- acceptance

By assigning the statements in the specification we can count the number of requirements, the number that are functional, and assign groups of them to particular teams.

4.3 What is Domain Knowledge?

Use cases and classic context diagrams used in Yourdon and SSADM all have a focus that is somewhat too narrow, the focus at least in the business or user specification is on the boundary, rather than on the workings of the machine or system, which is premature design.

First of all what is needed is an understanding of the domain problem to be solved, the application domain in which it exists, the domain objects that populate it, and their behaviours, relationships, and rules. Not just the pure system to user interface typified by a classical context diagram or a use case. A broader picture is needed. This allows better descriptions of the phenomenon that must be shared at the system boundary between the domain objects and the system (machine) to be developed. The full understanding of the fit in to the environment constitutes the domain knowledge.

Objects in the domain are not under the control of the project or the solution so part of the domain analysis establishes the system boundary. This process may raise issues about whether it is appropriate to negotiate the boundary or become a stakeholder and feed into another project. Often business procedures may need updating but not by the project. Rules can implement legal or sector directives and regulations.

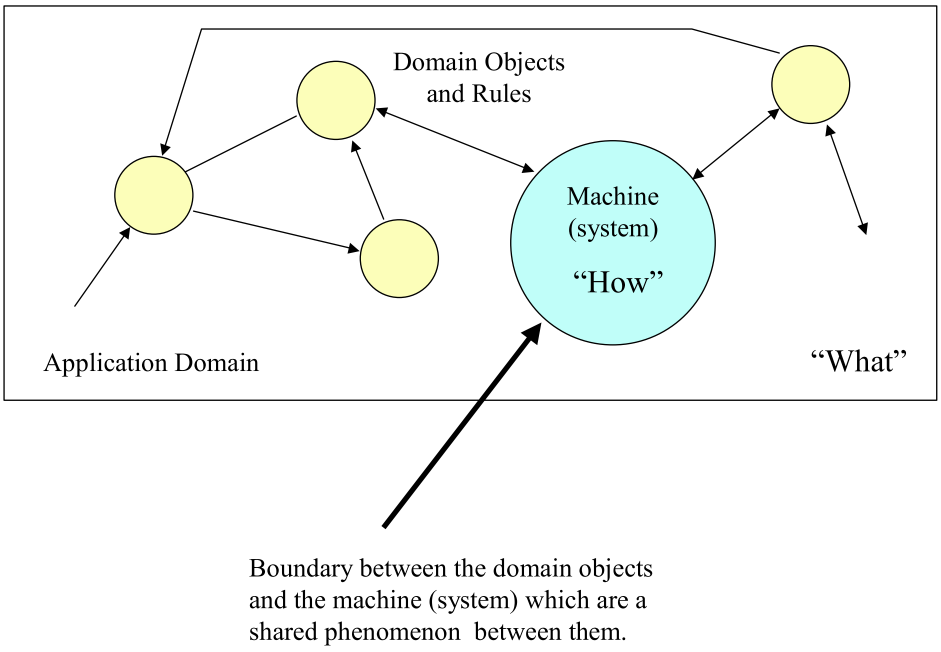

Domain Objects and Rules
Machine (system)
"How"
Application Domain
"What"
Boundary between the domain objects and the machine (system) which are a shared phenomenon between them.

Figure 1 - Domain Terminology

The reason why a system needs to behave as it is described is because of the domain, in to which the machine must fit in, not just in terms of the interfaces over the boundary but the overall behaviour of the wider system. Typically domain information is a description of the overall operating environment. More specialised domain knowledge includes reasoning why certain behaviours are needed or not needed, for example;

- A requirement placed on a speed control system when specifying a train, might say for example, that the train must not exceed 60 km/hr, because of the last position where a driver would see the red signal and the distance to the red signal. If we do not exceed 60 km/hr when we first sight the signal we will be able to stop in time.

- However the true reason why its 60 km/hr is not actually stated. Investigation identifies the braking distance depends on a number of factors including braking efficiency, train mass, weather conditions, rail adhesion, and driver reaction times. If for example the train was lighter or had less efficient brakes then braking distance increases. So the train might not exceed 60 km/hr but still not be able to stop at the red signal. While the machine might be developed according to specification the specification it self would have been wrong.

- The factors affecting the braking distance are not characteristics of the software speed control system but of the domain objects and rules that they obey. For the software to be work correctly it must be consistent with these other factors. Braking efficiency needs to be stated as domain knowledge related to the brake object and the reaction time of the driver to start applying it. These conditions must be proved to be true to satisfy the argument. Under certain conditions the normal braking mechanisms are not true, aquaplaning of wheels means braking is not applied, and additional domain knowledge must be added to complete our understanding.

- A traffic light crossing system specification typically says things like a pedestrian will push a button and cars will slow down when confronted with an amber light and stop when the light is red, the pedestrian crosses the road on receiving a green light, and after a period of time will return to red, and the driver given a green light. Safe usage following these rules occurs not because of the machines functionality, but because of domain information, formulated in the high way code about responses to the colour of traffic lights, and the elapsed time from an amber to red light will need to take account of the applicable speed limit and driver reaction time, and the time the lights are red will need to take in to account the time that it takes a pedestrian to cross the road. For a system or machine to work and meet the users need, it must obey the domain knowledge, and the domain knowledge must be proved true, or an incorrect solution will be specified.

While the above are simple and obvious examples, similar problems are found in many systems, where the domain objects are not fully understood. Relevant domain knowledge may apply even though the domain object is not directly linked to the system or the machine. This is why care is needed when using context or use cases as the starting point and not the broader operating environment. Notice the focus is on describing the domain application and the problem, it is not about an abstract boundary, nor does the specification focus on the design of the system or machine its self. That comes latter.

As we have said, each piece of domain knowledge must be satisfied for the boundary specification to be true, and changes to the domain must be managed and reflected in the specification if the system is still to behave and perform correctly, that is, still share the same phenomenon. We need to carefully describe the domain and look for domain issues of the types outlined. This is how we start to establish our requirements. At a practical level this is done with existing system studies, reviewing procedure manuals, interviews, use case sessions to name a few techniques, but domain analysis is how the data is then evaluated.

Knowledge of how the solution is to work in the domain environment is necessary, as timing, information, and process sequences will depend on seeing the big picture. Often use cases may output information to the domain while another use case receives an input. The use cases are often treated independently yet in reality these are related via a domain process. Use cases must be fitted into business scenarios to understand such relationships. It’s worth noting what also needs to be done to prepare this domain environment to operate and maintain the new solution. We often see well written specifications provided to a supplier but overall no specification details what needs to be done with in the organisation, such as training or changes in the wider process context, perhaps altering certain other databases or equipment items, or rules so that the benefits of the solution can actually be delivered.

We may also need to define the problem and use techniques to fully elicit the problem, by using methods such as soft systems method and systems failure method, or simple baselining of existing performance and analysing the organisation, application, infrastructure, and paper systems to establish the current problems, limitations, and issues, and identifying bottle necks, drivers, and inconsistencies. Complemented with an analysis of the market, the organisations or products current positioning, the market needs or trends, and needed business returns it is possible to define the overall objectives or reasons the project is needed in the first place.

The approach to good requirements specification is based on the domain application, problem framing, shared phenomenon between domain objects and the machine, its specification and its design, and we heavily draw on the work of Michael Jackson. If domain knowledge changes then the solution is likely to have to change as well. So monitoring domain knowledge is also as important as the machine specification its self.

4.4 What is a good requirement?

What makes a good requirement is difficult to define. However we can test a requirement for certain properties which would indicate it is likely to be well formed and means exactly what it says. Examples of tests are given in Table 4.

Table 4 - Test attributes of a good requirement

Group	**Test**	**Description**
Individual Requirement	Atomic	A sentence or paragraph only includes 1 requirement. Often a paragraph mixes several requirements, making testing and traceability difficult, and in some cases introduces an implicit requirement.
	Realisable	That the requirement is achievable with in the time scales, costs, and technology constraints, for example ensuring that the requirement does not call for a perpetual energy machine, to be delivered in 2 weeks.
	Testable	That independent tests can be constructed to prove that the requirement has been achieved or not achieved. This often means that the test is quantified in some way. A draft acceptance test should be formulated before the requirement is released.
	Unambiguous, specific and complete	That only a single meaning can be construed from reading the requirement and it means details or the precision or accuracy that may be needed.
	Does not prematurely dictate the design or gold plated the requirements or design.	That the requirement delivers only the higher level requirement, it does not impose an unwarranted constraint on the design, architecture, technology or operation. Gold plating means the design is over specified.
Document or Requirement Set and to the higher set of requirements	Non-conflicting	These 3 attributes are similar. Each requirement must be tested against itself and against each of the others to determine if they duplicate each other, conflict, or over-lap in some way. This can be achieved by take a set of requirements in a given document pasting them into a 2 * 2 table as both the column and row items.
	Non-over lapping	Each intersection is then examined. If a problem is found it is flag by entered a value or code in the cell. Sometimes over lapping requirements might indicate a level of restructuring is needed.
	Not duplicated	These are then managed until resolved by modifying either or both requirements

Group	Test	Description
		through rewriting the requirement or negotiation with stakeholders or designers. The matrix also provides evidence for the requirements or design review that this critical if slightly onerous task has been completed.
	Traceable, complete, and appropriate	Starting at the higher level requirement, see which lower level requirements or design elements it can be traced too. Review if they are appropriate traces, and collectively determine is the lower level traces adequately explain how the higher level requirement will be achieved. Where they trace to several requirements ensure they are compatible and consistent. Finally check each higher level object traces down and each lower level requirement or design element has at least 1 upward trace - this makes sure that no orphans exist.
	Consistent terminology	Where domain or specialist terms, names or documents are mentioned, ensure they are used consistently. This is best achieved by having a separate module of document containing a glossary, abbreviations and definitions, and links to each instance of occurrence. This also supports maintenance of the document. Without such a glossary and with a large specification confidence may be low that the terminology is consistent.

We also can analyse existing documents or verbal requirements for quality. True requirements are prefixed with the words “shall” or “will”. If the words “may”, “could” or should” are ambiguous and legally may not be viewed as being a necessary characteristic of the deliverable. Similarly where words such as “and” and “or” are used the statement is often a molecular requirement, one containing several requirements. Good requirement statements only have one subject which makes testing easier. Words such as “TBA”, “TBD”, and “for example” indicate the requirement statement is not complete which often leads to debate about what is really required. Until all information is recorded testing is not possible.

You can identify these phrases using word processor searches or filters in specialised requirements tools. Keep a record of how well the specification meets these criteris.

Use them as part of the review process, and monitor if the specification turned out to be problematic at a latter stage because of lower criteria.

Terms such as “may”, “should” or “could” can be seen as desirable properties of the system, rather than mandatory. But make sure that is what the stakeholders and users think and agree too. Some times a requirement is implied by use of “should” for example. Where it is actually a mandatory need this type of error needs to be corrected.

We also look for requirements that are not well defined or capable of unambiguous testing on deliver. More about this part of the process later. By assignment of a information type we can quickly filter down to the requirements, and look to restructuring a specification, so similar information is related together, while still retaining the context of the full document. It also means we know what the requirements are and how many there are. And we can improve the quality of the document.

4.5 Attributes

In the previous section we introduced the word “related information”, that is each of the requirement types defined above, can possess related or additional information. This related information helps us to make sense of the requirement and undertake management tasks. We call this information requirements attributes. An attribute could be the information type, secondary categories, or indicate the quality of the requirement following review.

As projects develop we collect further information related to the “requirements”. This information is the candidate pool for attributes, many of which will relate to a stage in the requirements engineering or development process as shown in Figure 4. Some attributes might actually use a reference, point or links to other information that holds the detail. This will depend on the information or data model you have decided to use, whether that’s a set of word or excel documents, or a relational database. Attributes can be used to establish points for sorting and counting on for reporting purposes. Examples of attributes are shown in Table 5 as follows;

Table 5 - Requirements Attributes

Process Stage	Attribute	Purpose and / or enumerated options under attributes
Management	Unique Identifier	To provide a stable non changing reference. Problems can occur when maintaining documents when the identifier is the automatic heading number, for example when requirements are added, moved between sections, or removed.
Elicitation	Information Type	Not Defined, Requirement, Context Text, Heading, Domain, Definition
	Requirement Type	Not Applicable, Capability, Non-Functional, Process, Constraint
	Secondary Type	Performance, Reliability, Maintainability, Security, Safety, Environmental, Portability
	Key Feature (or linked)	Y/N - Details in rationale. Used to quickly identify the key features or benefits that a system provides to ensure focus is maintained on them, and filtering can be applied.
	Domain Knowledge Record Details	Often referenced or linked to a file of domain knowledge and rules.
	Domain Knowledge Checked	Applies to requirements market domain knowledge and indicates that the domain knowledge has been signed off as correct.
	Owner or Stakeholder	Who needs this service or imposes a constraint. Maybe a named individual, a regulation, or business rule. Stakeholders who has an interest in the requirement. Often 2 owners are needed - one who requires the service such as the user and one responsible for delivering it to the user, such as some one in the project or design team.
	Source Data	Reference to the owners memo, meeting, or use case sessions for example when raised It records how the requirement was captured and the reference to the full source material.
	Rationale	Records why the requirement is needed, some times this is not always obvious. Some implementation will use this attribute to point or link to a file detailing the logic.
	Guidance / Notes	Standards, methods, technologies, that may be used to satisfy the requirement.
Testing stakeholder requests or requirements	Individual Requirement & Comments	Tests performed on each individual requirement, for examples of enumerated or multiple list items that might be used in this attribute see Table 4. The problems

Process Stage	Attribute	Purpose and / or enumerated options under attributes
as they are formulated.		needs to be recorded and a text comment recorded describing the problem. This split allows counts of problem types to be made as well as producing and resolving problem reports.
	Requirement Set & Comments	This is applied as the document level rather than each requirement. See the Table 4 for examples of enumerated or multiple list items that might be used. Again the problems need to be recorded and a text comment recorded describing the problem. This split allows counts of problem types to be made as well as producing and resolving problem reports. Remember to reference both requirements,or maybe more, in each statement where duplication, overlap, conflict and inconsistencies are found so they can be correctly corrected and no problem is lost.
Planning and risk management (some attributes can also be used in tender assessment along with attributes from following V&V processes)	Assumptions	Does the requirement or result (if used in V&V) contain any assumptions?
	Issues	Are they any issues associated with the requirement, domain knowledge, or result, for example while the requirement is true for now against the domain knowledge, the domain is likely to change.
	Satisfaction Factor	How happy will the owner or stakeholder be if this requirement is met? Delighted or minimal expectation. Helps on prioritisation with the dissatisfaction factor. They may not be reciprocal.
	Dissatisfaction Factor	How unhappy will the owner or stakeholder be if this requirement is met? Very upset or not really fussed. Helps on prioritisation with the Dissatisfaction factor. They may not be reciprocal.
	Priority / Release Date	When is this requirement needed? Important for prioritising the development of features in an iterative program.

Process Stage	**Attribute**	**Purpose and / or enumerated options under attributes**
	Risk Factors	How complex is the requirement or the solution? This attribute is stage development. How mature is the technology used in the solution and in the intended domain? Does the development team **and** the supplier organisation have the appropriate capability maturity level for the domain and technologies to be used.
	Risk	What is the risk it can not be achieved, could be a number, or a description, often has an accompanying commentary. Use information collected on the previous risk factor attribute and the quality tests (e.g. how complete is the requirement, is it poorly worded, or not readily testable as currently written). More structured approaches will break this down in asking what is the consequence of not achieving it, the likelihood of this happening, and take into account the issues and assumptions listed under other attributes, and evaluate the complexity of the requirement and how well the requirement is understood. Other wise use the information in the other attributes and make a qualitative assessment. As a check rank the requirements in risk order and run a sanity check to see if the ranking seems appropriate given the information available.
	Benefits	What business benefit does this requirement contribute to? This applies to requirements only, and not information that is a test result or domain knowledge for example.
	Work Planning	In order to test this requirement or decompose it further what work is required? In more sophisticated processes time scales, work streams, status and other information would be recorded as well.
	Status	None-accepted, decomposed, formulated, tests, approved. We might add the review date, reference to meeting notes
Design Review	Functional Allocation	In a system or product specification the attribute identifies either where the function has been allocated to the product structure or what functions are in each product. Often additional review and comments attributes are added.
	Design Reference	The configuration of the design, hardware, software. Any FCA or PCA references.
	Design Review Factors	Technologies to implement functionality, performance, security etc.

Process Stage	**Attribute**	**Purpose and / or enumerated options under attributes**
		• Designability • Testability • Manufacturability • Install/Constructibility • Operability • Maintainability • Disposability • Design to Cost • Programme
	Design Review Comments and Author	For each factor consider to be problematic a short description and rational is give along with the name of the author. More elaborate versions may have actions and details of discussions, dates for design reviews etc.
Verification and Validation Planning	Negotiations or V&V Planning	Completed by the supplier. Is the requirement accepted? What testing is planned? It is advisable to use additional attributes to indicate the stage of project to which any test will be applied.
	V&V Stage	Various V&V Method Types are likely to be applied to each requirement at different stages of the project. For example an multiple or enumerated list may contain the following life cycle milestones : RRR, PDR, CDR, FDR, TRR, FCA, PCA, Cut Over, Commission, System Acceptance, User Acceptance.
	V&V Method Type	Next to each requirement the means of testing is specified in advanced, for example using a multiple list or enumerated list, such as : Inspection, Analysis, Demonstration, Simulation, Trial.
	V&V Method Comment	Contains the details to complement the V&V Method Type.
	Waiver or Deviation Needed	Generally a supplier y/n perhaps with additional attributes for details, reasoning etc.
	V&V Results Summary	Contains a summary of the result for a specified test at a specific stage,
	V&V Results Source	Points to the file(s) containing the results, and may also holder information on the

Process Stage	**Attribute**	**Purpose and / or enumerated options under attributes**
	Document(s), File reference etc.	configuration versions of the software and hardware, against the agreed specification baseline.
Verification and Validation Evidence Submission and Review	V&V Trace Object	Y/N - Tests performed on each individual requirement, for examples of enumerated or multiple list items that might be used in this attribute see Table 4. This might be implemented by a tool or paper, using appropriate guidance.
	V&V Trace Suitability	Y/N - As for V&V Trace Object.
	V&V Trace Comments	The reviewers name and comments on the trace and suitability. Addition attributes may be added, to contain details of actions that need to be addressed to correct any problems found.
	Negotiations or V&V Planning Comments and Author	A review of the proposal by the acceptance authority.
	V&V Planning Acceptance	A review, perhaps at the client, examines the planned V&V activities proposed by the develop. Addition information may be added on any problems found and discussion to reach agreement with the developer / test organisation.
	V&V Review Comments and Author	A review of the V&V evidence by the acceptance authority
	Waiver or Deviation Reviewer Status	The reviewer marks any request, indicating whether the application has been accepted or rejected, and details of the progress through the deviation or waiver process.
	V&V Status Summary	Generally an enumerated field to indicate how far through the process the requirement has progressed e.g. no supplier response, negotiation acceptance, negotiation acceptance, compliance planning submitted, compliance planning reject, compliance planning accept, compliance results submitted, compliance results accepted, compliance results accepted.

You may wish to add attributes for reviewer requirement or various status indicators relating to the process stage that currently applies. Other typical information to hold is who is the attribute owner and who has access to it. If you are using a software tool this information can be used to set read, write, and change permissions to the attributes.

Other attributes may be used to implement a Goal Model in a document or requirement module that forms part of the requirements tool or database. Attributes may be used to support reporting, searching, or filtering capabilities, for example a date based rule may generate an attribute to indicate a changed requirement from a given baseline for impact and trace analysis. Many requirements tools, as discussed in Section 11 may be configured to automatically do these type of activities.

4.6 Monitoring progress through the Process.

Many attributes relate to improving the quality of the requirement set or have an objective of identifying certain errors. For example, conflicting or overlapping requirements, the rational on why the requirement is present, or is consistent or is part of a definition to be contained in a linked glossary. Such attributes can be used to sort and filter on to identify problems or to identify candidates that need to be linked or related - remembering of course to record the reason for the relationship.

We also record information on the status of the requirement, domain knowledge, or test with respect to testing or validation at various stages of the project. This may relate to how far the progress it has progressed, for example non-accepted, decomposed, formulated, tested, and approved, and accepted by the next level down. We might add the review date and references to meeting notes as well. Remember though some of this may become invalid as changes are made, in effect, it moves back in the process, requiring retesting, approval, and release and agreement to the next level down. Sample progress reports may include:

- Number and status of stakeholder needs progressed through the process
- Number of firm requirements traced to the design indicating potentially under design
- Number of design items that trace back to the requirement with no orphans indicating over design
- Number of requirements satisfied by testing of the design or product
- Number of changes proposed and number of requirements and product design items affected
- Are we still trawling?
- Still trawling and refining but have realised details to the next level below?
- Requirement formulated, tested, and awaiting acceptance from next level requirement accepted at the level below

Attributes are important, use of standardised types of classifications, with exact definitions allows searches and counts to be made. When reported and analysed this information allows effective management of the process to be initiated and for the risk and progress to be understood and controlled.

4.7 Types of Relationships Between Requirements

In Figure 4 you can see links exist between modules, most tools allows allow links in a module or specification itself, joining requirements. We can name these links to aid understanding and assist in filtering, simplifying the way we trace our way through the requirements model. Types links includes;

Table 6 - Types of links and relationships between requirements

Type	**Description**
Satisfies	Shows how a lower level requirements implements a high level requirement such as the planning and evidence that is gathered in the V&V / testing process.
Derives	Records decomposition logic from higher to lower levels often within a specification and between specifications.
Conflicts	Between requirements within a document or between documents.

Relationships can be shown using manual traceability tables or graphically with lines connecting the requirements. Hyper links are a form of traceability. Other examples include matrices showing requirements cross references or conflicts, requirements to the architectural design, requirements to domain knowledge, and requirements to stakeholder needs. They summarise information to specific view points. The requirements will often exist in different documents, for example the user and system specifications, and the test or acceptance specification containing the results. In the next section we will look at the different types of documents that are often use and how they fit into the process and relate to each other.

4.8 Information Models

We can know put together some of these concepts to create an information model which defines:

- A set number of documents, modules, files or references which contain similar types of information
- relationships between these document, modules etc
- a set of views which use information from a given document, module and can use related information from other documents or modules. The views can be used to package the information on screen, create reports for printing etc. Views can be used to implement steps in a process, such as creating, reading, updating

and deleting information and adding, reading, updating and deleting associated information, for example a requirement and its rationale as an attribute to contain a review of the requirement and where it satisfies its parent.

Information models are often visually represented as shown in Figure 2.

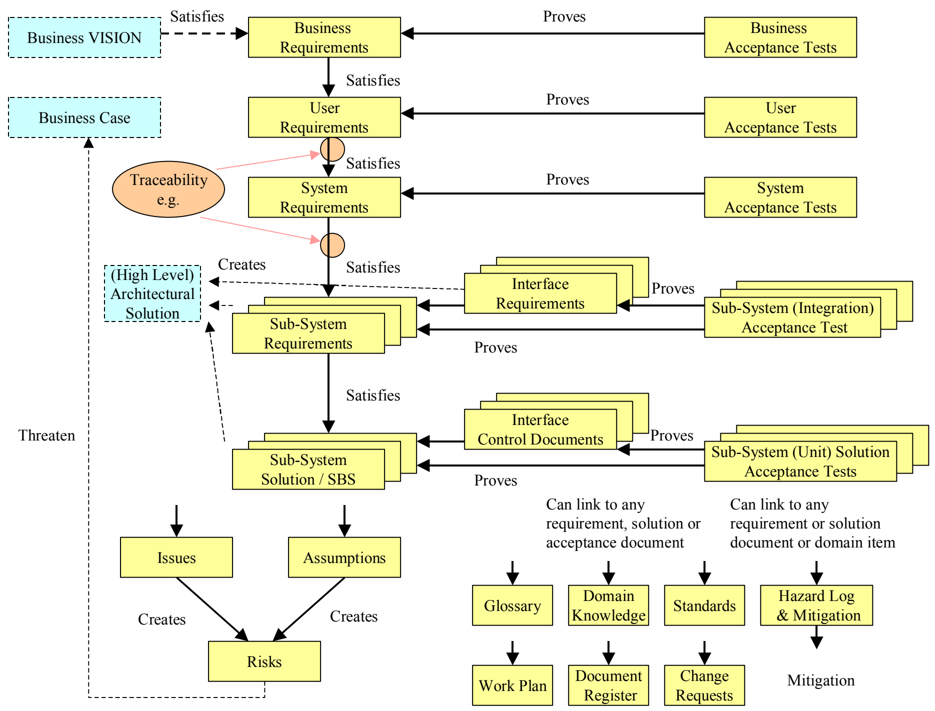


Figure 2 - Sample Information Model

5. The Requirements Specification

5.1 Specification Structure and Contents

Having defined a requirement, we need to provide some structure, ideally to minimise or eliminate duplications, in a document that is easily navigable, and importantly contains the right information to help the analysts, developers or suppliers at the next level of development.

In fact outlining the structure and contents of the specification document is one of the first steps, before even thinking of the attributes, requirement types, or testing. The requirements specification will need to contain sufficient information to enable the developer to produce a solution that actually meets the stated needs - assuming of course this statement of needs is complete, and can be correctly interpreted.

We can use several requirements templates as a starting point. Consider each section, determining if its necessary, and if so, undertake some work to capture and refine the appropriate information.

Table 7 - Specification Templates

Sample	**Comments**	**Web Reference**
IEEE 830	High Level Outlines of a software requirements specification. Standard and modified versions can be found at:	www.mtsu.edu/~wahl/csci470/gradwork/ieee_830.htm www.dur.ac.uk/~dcs8s00/phases/require.html
MIL STD 498	While this standard on software engineering processes has officially been withdrawn, and replaced by IEEE/EIA 12207, its accompany Data Item Descriptions for a range of specifications is still an excellent set of templates. Key specifications include Operational Concepts Specification (OCD), the System/Subsystem Specification (SSS) , Software Requirements Specification, and Interface Specification (IS).	http://sepo.nosc.mil/Standards.html
ESA	European Space Agency : This guide includes template User, System, Interface, Test and other specification types as well as guidance notes.	http://sting.web.cern.ch/sting/ESA.txt
VOLERE	Combines user, system, and management details. Comprehensive but may benefit from being separated into elements to match the business, user, and system views, as overall it contains more information than any one actor at a given level would need.	www.atlsysguild.com

Not all requirements are held in the same document, instead they separated out based on the point in the development life cycle they are initiated and their objective. each contains only the sufficient information to support its purpose and specify the means by which it meets any higher level requirement. This is reflected in Figure 4 which outlines specifications at the business, user, system, and test level.

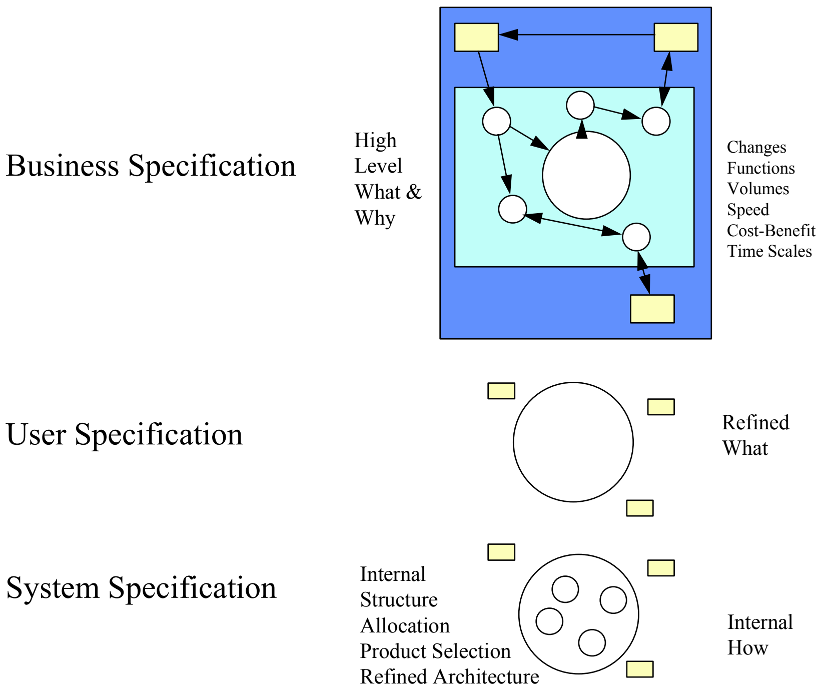


Figure 3 - Specification Focus & Analysis

Splitting up modules has other advantages, importantly in a collaborative working environment, to simplifying simultaneous usage, and baselining of any particular version for use by other groups at the next level down, and the incremented version is then free to be developed further or to modify requirements in response to changes. In addition to the documents shown in Fig additional information maybe held and related to requirements and attributes, or be structured in a similar way to an RDBMS. An example of an information model for operating at the enterprise level is given in Fig. 2.

Specification or Document Type	**Objective**
Business	Vision and scope
User	Needed boundary needs, the users, and interfacing systems.
System or Functional	The internal high level design to deliver the use needs within the defined constraints.
Supplementary	Some methods, e.g. RUP keep non-functional and management requirements in a specification associated with the system or functional specification.
Interface Specifications with attributes on integration testing and acceptance at either the higher requirement level or at the product level.	At the system level a number of subsystems may be defined, or a design document may need to defined the boundary interfaces in detail. Interface specifications address these issues. For example consider 3 subsystems (A, B, C) that must all work together, in this case 3 interface specifications (AB, BC, AC) will detail the

Specification or Document Type	**Objective**
	interface requirements and drive the interface test requirements. Progressive levels of integration testing will exist in between although this is often dealt with by adding test requirements for these phasing and monitoring the progress.
Test Specifications with results and reviews stored as attributes	V&V or testing. These take the requirements, for example from the user and system specification, and use the initial "testability" quality attribute, and prepare detailed test cases for each at a planning stage, detailing at what project phase and using what method. Then record the results. In most projects the planning and results will each need to be approved by an accepting authority.
Management / Commercial	Most contracts will also have associated management and commercial requirements which are often held separately. These include reporting requirements and arrangements, terms and conditions, project management and organisation, milestone details, and process requirements and constraints.
System Breakdown Structure and Application with attributes on site inspections and acceptance	While not a specification as such a useful document to include is the system breakdown structure or products listing. This can be useful to; • document functional allocation • generate physical interface matrix • document physical applications and architecture, where numbers of items can be related back to target costs. The SBS will often be used to drive the Work Break Down Structure through design, make, install, commissioning, and operate phases. It allows links to be established with the system specification to track functional allocation to ensure each function or realised in a product, its version etc, and ensure every products function is also needed. Inclusion of SBS information allows full traceability and support configuration management. A separate model may map the SBS to locations, indicating the number of items and specific interfaces in each location. A corresponding compliance module can be kept for each generic product and site installation.
Glossary	Definitions and abbreviations, particularly those with commercial or acceptance implications, or terms that are

Specification or Document Type	**Objective**
	used in a number of documents. This approach helps to ensure self-consistency between documents.
Document Registers with attributes of document reviews.	Often a compliance view or compliance (acceptance or V&V or testing results attribute) will reference out to a document containing the results and any out standing issue. Logging each document and where a claim is made can help in a number of ways. Often a single document will address a number of requirements. By listing the related requirements a check list can be fed into the documents technical review process to ensure they document adequately addresses the claim and no other information in the document contradicts it. The result of the document review is recorded in the register and the links back to the compliance view or matrix show the status of the claimed documentation. The documents listed in the register could also be linked to each other in successor and predecessor links and so form the basis on a simple document configuration management system when completed with version number, release note, document review, and observation attributes.
Risk, issues, assumptions and work planning items.	Each requirement or test or test result may raise issues or may be made on assumptions which translate to risks. In other cases to reduce the risk or remove the assumption, which could include a task to better define a requirement or establish certain domain or technology facts, tasks need to be undertaken. Monitoring what the tasks need to be done ensures none are left out or to ensure sufficient resource is available or flag resource that is needed or avoid resource working on tasks that are not authorised. This approach has benefits that is the work is not authorised then at least the inherent risk in requirement definition or acceptance is then explicitly logged.

Table 8 - Types of Specification

What we see is there is a layer of specifications and documents, each focused to a particular set of objectives, but the true value is the overall picture that is obtained when the information is seen collectively in different threads and views.

What the layers do is at the high level set a need or target, and the lower specification establishes a means of delivering that higher requirement. This may require some analysis or simulation, some functional allocation to physical products or determine an architecture. Through simulation or analysis we may test whether collectively the physical products can indeed meet the higher level performance or through put requirements. If they do then each products characteristics in terms of functional allocation or apportionment becomes the targets for the next level. Each layered pair sets a target and then shows that target can be achieved. Each test specification eventually has a results file showing ether the test has been met, along with reviewers comments.

In this way through the chain the overall business needs can be assured to be met and reported on at any stage showing overall progress, risk, requirement or product stability, and the impact of pending changes. It is now we can see how powerful good requirements management can be.

6. The Requirements Process

The process centres round the existence of a number of defined specifications ; business, user and system together with appropriate test specifications or V&V modules. They may be in a single document or module or split. Each approach has its strengths and weaknesses but ultimately this is a decision that each project will need to make.

Once the information model as described in the previous section has been agreed the rest of the process implements a set of steps joining each specification together. A simplified process and information model structure is shown in Figure 4 as an example. The steps of the process are based on;

- If at the user or system level testing the "requirements documentation" flowed down or collected from stakeholders and the users are analysed to determine what are the actual sub-set of requirements, context, definitions and to determine if the requirements are accepted, are understood, and meet good requirements practice. Often additional attribute information will be needed.
- Next is decomposition and trawling to develop a means of delivering the higher level requirement, this will often involve using some form of logic, or goal based reduction technique, linked to any relevant domain knowledge. Various activities will need to be performed or if being performed by others, liaison established and the relevant information extracted in a controlled manner. Examples include;
    - ◊ At the business level stakeholder needs will be built up using business market, SWOT and similar analysis to establish a VISION and scope statement, which are then tested, traded off against each other, until initial business requirement's are established.
    - ◊ At the user level use case and scenario analysis, boundary performance, security, and safety analysis at the boundary
    - ◊ At the system level, system level uses cases may be performed, as well as converting the user level use cases, through an analysis model, into object

models, sequence diagrams, and allocation to candidate infrastructure assets, internal timing and volumetric work, and failure analysis work performance to determine if the solution will meet the user requirements within the defined constraints.

- When the team at the level below receives the higher level specification that has been flowed down requirements, they provide an initial compliance statement, in the form of a V&V or Test Specification, detailing the tests that will be applied at each stage to show how the higher level requirement has been met. This is then reviewed by the team at the level above for acceptability and offers confidence that the lower level specification will meet the higher level requirements.
- When the requirement is considered to be well formulated it is tested to ensure the requirement is well formulated, is sufficient and does not excessively meet the higher level requirements, and is consistent with other requirements and terms used. They will be examined in a requirements review.

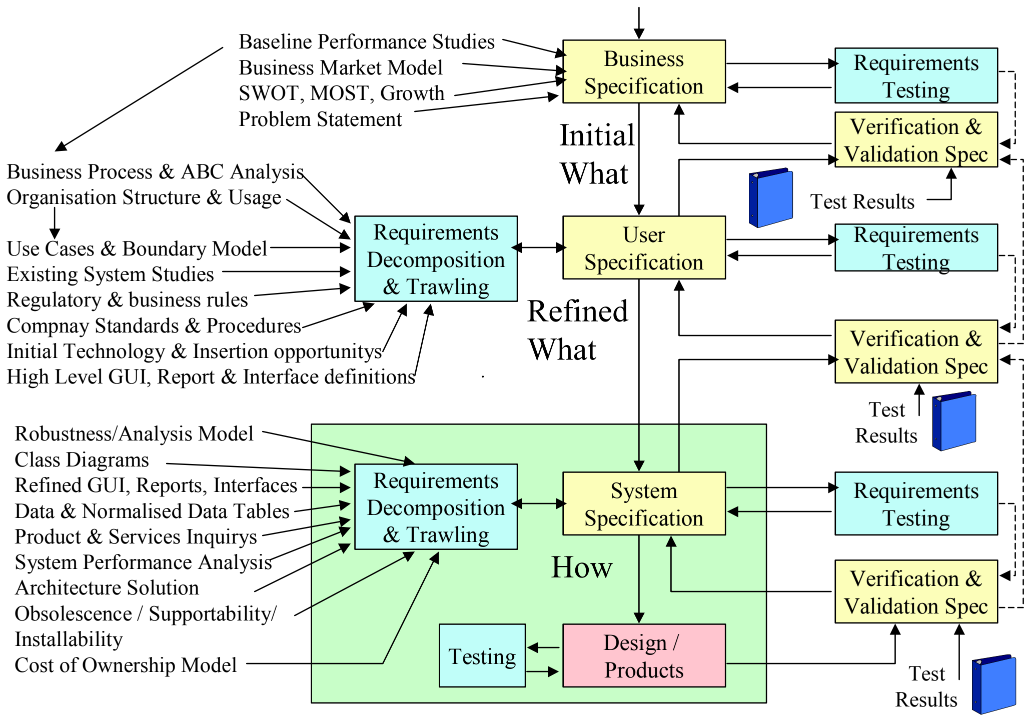

Baseline Performance Studies
Business Market Model
SWOT, MOST, Growth
Problem Statement
Business Specification
Requirements Testing
Verification & Validation Spec
Initial What
Test Results
Business Process & ABC Analysis
Organisation Structure & Usage
Use Cases & Boundary Model
Existing System Studies
Regulatory & business rules
Compnay Standards & Procedures
Initial Technology & Insertion opportunitys
High Level GUI, Report & Interface definitions
Requirements Decomposition & Trawling
User Specification
Requirements Testing
Refined What
Verification & Validation Spec
Test Results
Robustness/Analysis Model
Class Diagrams
Refined GUI, Reports, Interfaces
Data & Normalised Data Tables
Product & Services Inquirys
System Performance Analysis
Architecture Solution
Obsolescence / Supportability/ Installability
Cost of Ownership Model
Requirements Decomposition & Trawling
System Specification
Requirements Testing
How
Verification & Validation Spec
Testing
Design / Products
Test Results

Figure 4 - Basic Requirements Process

- The requirements, either singularly or as a set in a baseline document, are then flowed down to the next level of the supplier chain, where negotiation may then commence on whether the requirement is understood and accepted by the team below, with the process then restarting as described above.
- The lowest level in the supply chain evolves a design or further specification which delivers the flowed down higher requirements and progress is established at periodic product Design Review's. An import step is defining how each requirement will be tested - this is the Compliance Planning process.
- As the specification to design progresses, at the defined life cycle points the agreed V&V or Test Plan from the earlier step, is implemented. Test results are

collected and submitted for acceptance by the higher level team as part of the Compliance Evidence Submission process. At the higher level the evidence is reviewed and tracking work is performed to show how each higher level requirement is fulfilled at the lower level, for example how the pre and post conditions in each user level use case are implemented in terms of properties, methods, and any data tables with the supporting infrastructure at the lower level. Specific evidence is also offered and reviewed to show that collectively, simultaneous object instances and data base read and writes can be completed within the overall performance requirements specified at the user level.

The basic process shown in Figure 4 needs to be broken out further to reflect negotiation, review and acceptance activities between each level and either requirements reviews or design reviews at the appropriate levels, where the relevant product would be examined, with evidence of process compliance and tests to show the product is fit for purpose. An out line of such a model is presented in Figure 5.

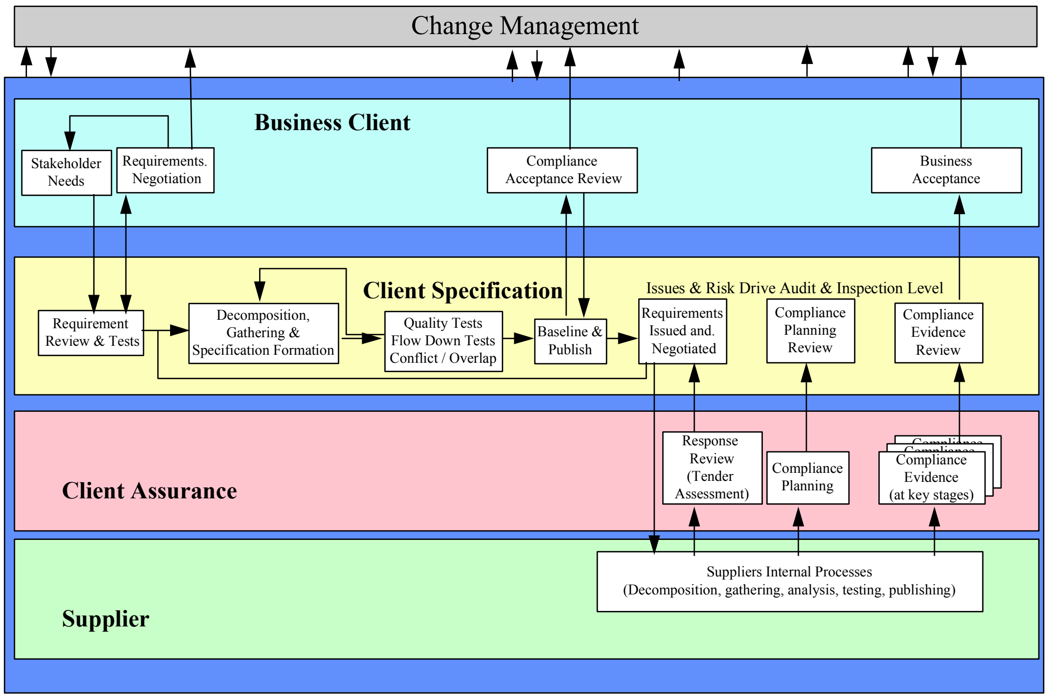

Change Management
Business Client
Stakeholder Needs
Requirements. Negotiation
Compliance Acceptance Review
Business Acceptance
Client Specification
Issues & Risk Drive Audit & Inspection Level
Requirement Review & Tests
Decomposition, Gathering & Specification Formation
Quality Tests Flow Down Tests Conflict / Overlap
Baseline & Publish
Requirements Issued and. Negotiated
Compliance Planning Review
Compliance Evidence Review
Client Assurance
Response Review (Tender Assessment)
Compliance Planning
Compliance Evidence (at key stages)
Supplier
Suppliers Internal Processes (Decomposition, gathering, analysis, testing, publishing)

Figure 5 - Basic Process Steps

A special case in the process definition concerns the use of scenarios and prototypes which serve to both assist in eliciting requirements and to validate previously captured requirements. This works because people when presented with a prototype of scenario walk through will tell you that’s not what they meant or wanted because of factors x, y , and z. So you elicit further requirements or refine existing ones until they can validate the set is what they need. So this activity joins the elicitation and validation processes.

Figure 5 shows us a high level view of the process. Next we need to consider a typical sub-process as show in Figure 6 below.

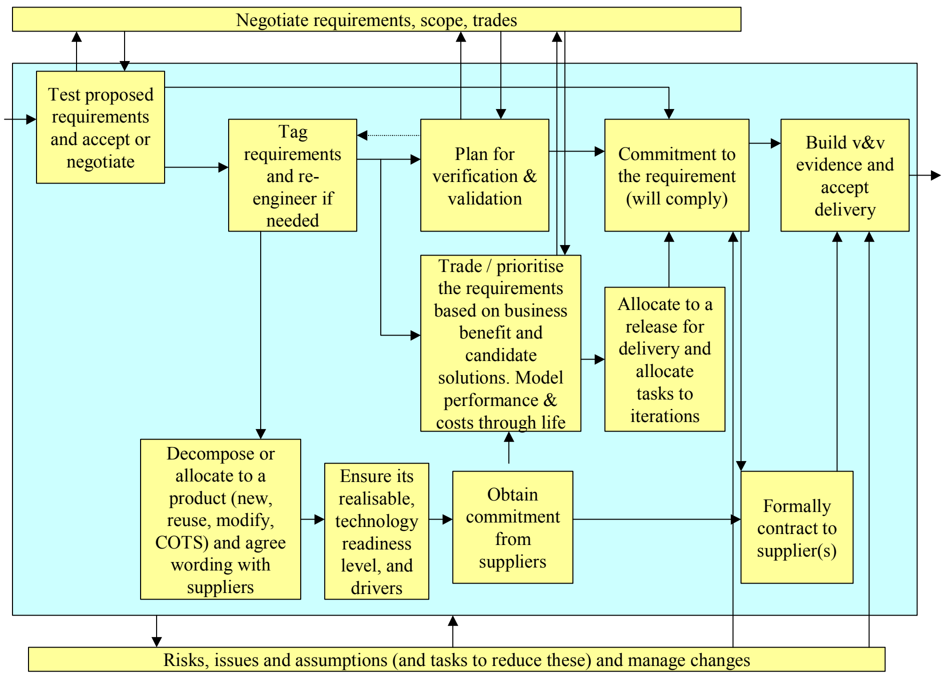

Negotiate requirements, scope, trades
Test proposed requirements and accept or negotiate
Tag requirements and re-engineer if needed
Plan for verification & validation
Commitment to the requirement (will comply)
Build v&v evidence and accept delivery
Trade / prioritise the requirements based on business benefit and candidate solutions. Model performance & costs through life
Allocate to a release for delivery and allocate tasks to iterations
Decompose or allocate to a product (new, reuse, modify, COTS) and agree wording with suppliers
Ensure its realisable, technology readiness level, and drivers
Obtain commitment from suppliers
Formally contract to supplier(s)
Risks, issues and assumptions (and tasks to reduce these) and manage changes

Figure 6 - Subprocess model for each requirement / specification layer

We have already discussed the concept of levels of requirement specification and corresponding acceptance or verification and validation document. The process in Figure 6 can be seen as recursive which receives information from the parent level and reports back up, while flows down and gains acceptance at the lower level, and in return receives reports back from the lower level, and final products for integrating up to the top level.

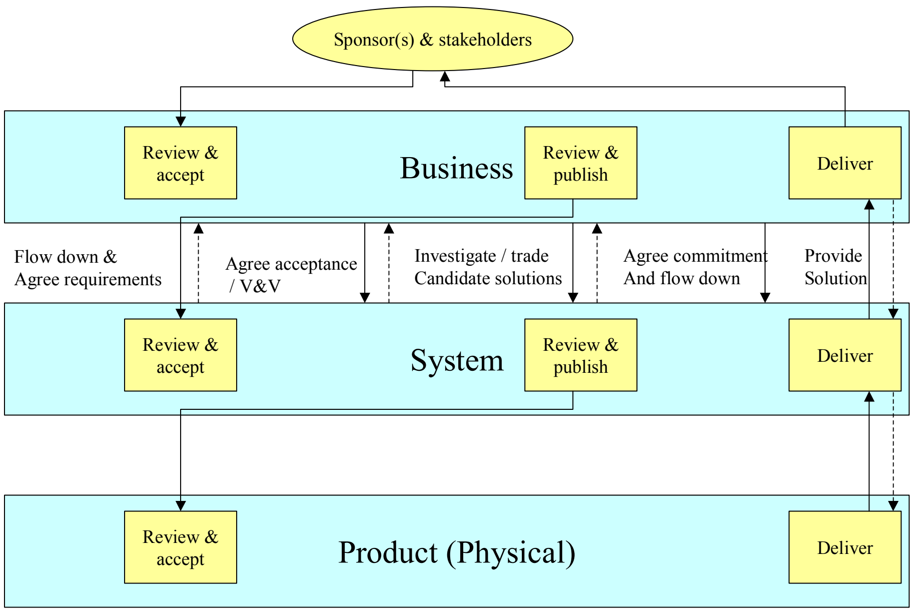


Figure 7 - requirements / specification layers

What is important is operating the process concurrently and iteratively, across all levels, otherwise if a waterfall or sequential approach is used then time scales will be pushed out and risk is not effectively managed. Use of iterations, with regular deliveries of prototypes, trials, and phased functionality, manages risk and shortens timescales. Strong requirements management, well co-ordinated, across all levels is fundamental to success. Use of requirements tools, either the same or capable of data exchange and using consistent information models or schemas.

7. Deriving Requirements, Goal Models, Rich Traceability and Verification and Validation

Having received a stakeholder need or a requirement passed down from a higher level specification it will be necessary to develop a framework for delivering that need or requirement. This will often require a strategy, further investigation into the issues and determining applicable domain knowledge, and structuring or decomposition to apportion additional requirements or design elements that will deliver the higher level requirement or need.

7.1 Deriving requirements

A derived requirement is one which is not readily apparent from a boundary function, but is developed to meet some internal condition, or through breaking down an existing requirement, and though investigation establishing a set of requirements. Therefore there will not be a source from a stakeholder for example but the rationale will be contained in logical satisfaction arguments or analysis. Two cases are worthy of a special mentioned ;

- Functional requirements such as this for reliability, safety or security are identified by analysing the requirement or functional model for example. Here the identified derived requirement is one that controls deviation or intrusion of the intended operation under particular scenarios, it is not delivering a primary service, but is ensuring the system stays within certain parameters.
- Non-functional requirements such as performance, reliability, timing, or throughput are derived through a process of apportionment from a high level requirement, simulation or argument. For example to deliver a percentage improved requirement by the business strategy revised performance figures over the existing performance will be needed. Where a functional design is developed and perhaps apportioned to a physical architecture it is possible to determine the individual performance characteristics of each element, and through simulation establish if the higher level requirements are satisfied. These characteristics become derived requirements.

In all cases the proposition is that a set of requirements and any relevant domain knowledge will satisfy some higher level requirement.

7.2 Goal models and rich traceability

A further method of deriving requirements is to use a tree stricture or goal model. The existing high level requirement is treated as the goal or objective and a set of domain conditions and sub-goals are identified. It is often useful to add information on barriers to the achievement of the sub-goal and identify further requirements that will over come them. Rather than merely recording the high level requirement and the lower level requirement, the rational or satisfaction argument is recorded as well, showing how and why they deliver the higher level requirement and so producing an audible record. This produces a rich and meaningful argument as to how the high level requirement will be achieved.

Often the rational has a logic that can include AND, OR, and NOT statements. For example out of 5 options, certain combinations may not be allowed, and a constraint has to be obeyed. Logic helps to clarify this and correct flow down requirements. Think of the rationale as a note pad to record what is often complex rationale. Merely linking 2 requirements loses valuable information and without it independent audit or peer review becomes impossible. Rather than direct linking, the decision logic, argument, or rational is recorded in a satisfaction document or module, capturing this information. Also each of these satisfaction argument steps may have related domain knowledge which need to be provided true and not to change during the development and implantation of the system without corresponding solution changes.

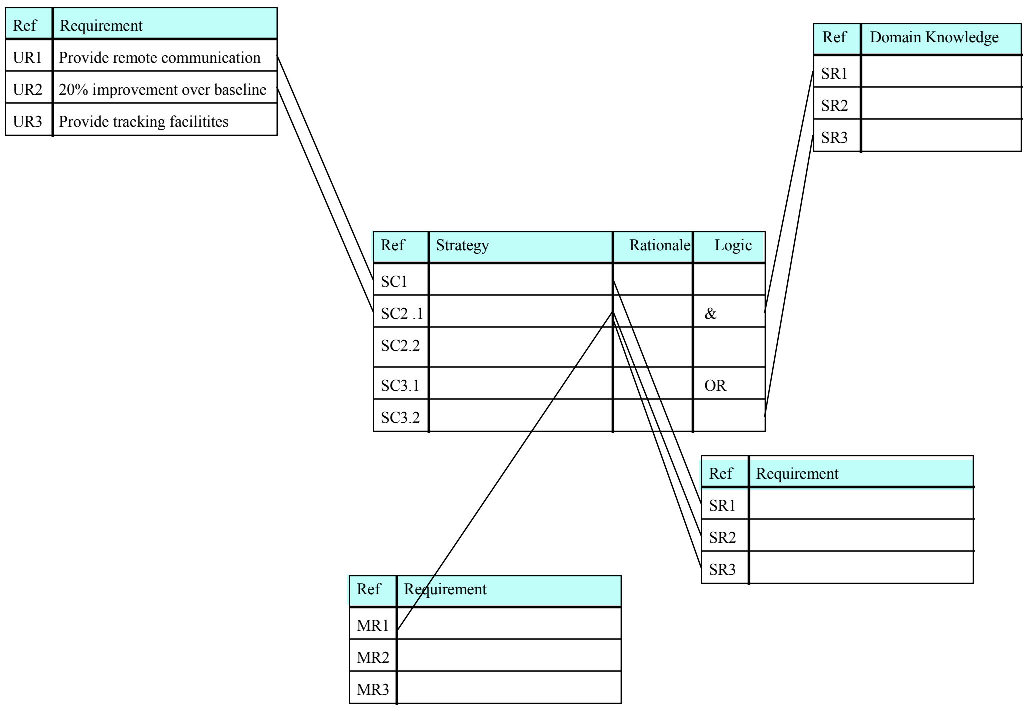


Figure 8 - Rich Traceability

Often the argument for delivering a higher level requirement or goal will not just be about delivering an application and infrastructure, it is likely to have organisational change requirements, publicity, or advertising, contractual and other softer requirements which equally must be achieved. This implies some allocation of derived requirements between a traditional requirements specification and other organisation action plans. Rich traceability allows this information to be viewed concisely.

Such structures can be presented, viewed, and commented on using rich tractability trees. These show the flow between the high level requirements, the intermediate rationale or satisfaction argument, any domain knowledge, and the derived requirements and logic. The flows can be supplemented with colour coding - terminal requirements an be red if they have not been decomposed further, green if it has, or blue it that is the end requirement. The flows can be code, say red for an unchecked or non-approved decomposition, green for approved, and yellow if it is suspect because of changes since a baseline. Collectively these techniques create power ways to examine the structured, relationship, and progress of the requirements decomposition and to build confidence that the high requirement has been translated into a lower level requirements, design elements, or a physical architecture.

This approach to rich traceability draws on Michael Jacksons work on domain and problem framing, and has usefully summarised by Jeremy Dick in the Summer 2000 edition of QSS News Bytes.

7.3 Use of analysis, behavioural and performance modelling and simulation to aid derivation and decomposition of requirements.

These techniques are used to confirm lower level requirement specification meet the parent requirement. In addition combination of model results and data exchanges are used to evaluate how emergent properties at the system level are satisfied by subsystem performance characteristics. For example in the rail transport system consider the interaction of the following subsystems to meet a system to provide a regular timetable stopping at multiple stations carry a given passenger loading;

- Track capacity in terms of number of lines and cross-overs
- Signalling systems to ensure safe separation and control movement
- Train power supplies to cope with acceleration and deceleration, especially abnormal operations such as start up following a perturbation, which places additional load on the system.
- Platform layouts and train layouts and door interfaces to ensure fast embarking and disembarking at stations to time table dwell times.

Each of the above models will need specific performance models, and the results of each need to be exchanged. In additional safety, RAM and security analysis will derive further requirements.

7.4 A Use Case Example of Requirements Derivation

While this paper is concerned with requirements, it does overlap with aspects of analysis and elicitation. Many specifications now express the functionality by using use cases, and activity diagrams. Use cases in particular though under UML should not so much be the diagrams, but rather focus on the accompanying textual description, using an outline similar to the following;

- use case level (business, user, system)
- actors
- intent
- displays, reports, interfaces, special formulas
- preconditions
- trigger
- steps
- post conditions
- expectation conditions relating to each step or minimal guarentees
- expected frequency and distribution
- volumes of data in each step
- alternative courses

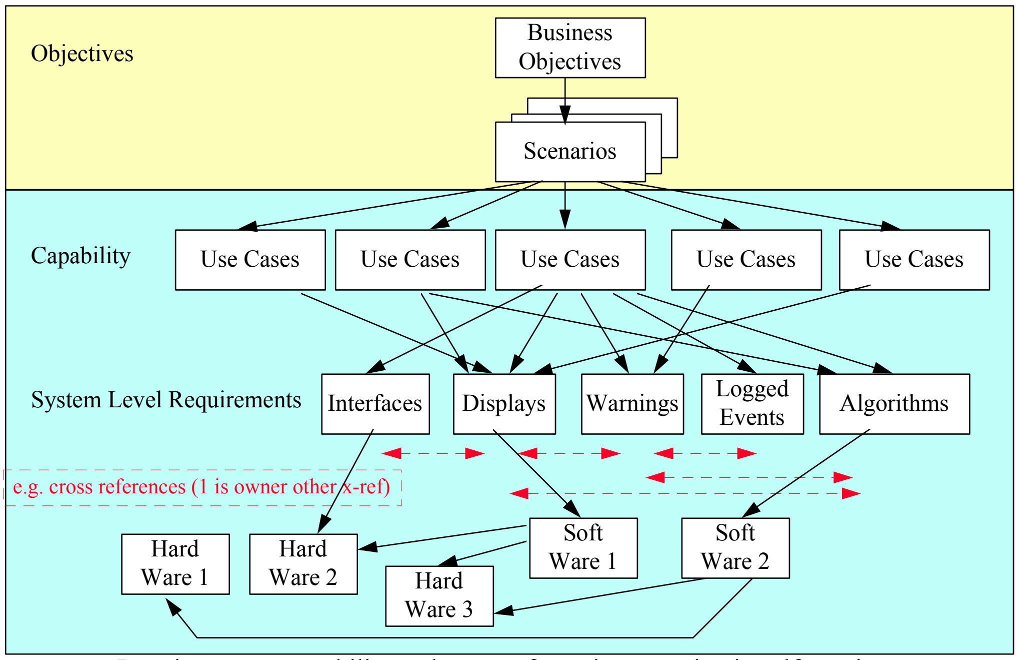
The use case describes the boundary interactions will still need supplementing with the domain knowledge discussed earlier. In a similar manner when we have defined the core “successful” operations functions, we work through the each step to see how the system would be have if;

- it could not be carried out
- its carried out early or late or changed accuracy
- ifs it occurs unexpected or out of sequence

While each use case is described, what must be well documented as the use cases delivering services to particularly actors. This information is best structured as a mix of headings and attributes to allow better management.

What is to be avoided is a set of actor centred use cases rather than the services to be delivered and the necessary actors that are involved. Collections of use cases allow particular scenarios to be serviced. A particular use case though could be used in a number of scenarios so implementing a level of reuse. The business scenario is process though is what’s important. Its easy to then see if the use cases in specification or at product acceptance deliver the needed services in the right sequence, with the right speed and so on.

Measuring acceptance at this level is easier than at the higher level. For acceptence testing purposes this traceability helps to build the test scripts which are developed from the use case scenarios (i.e. a specific thread through a success or alternative course) and collected into packages of use case senarios which deliver a business scenario. From this approach metrics on overall coverage of testing can be developed and focussed based on risk elements such as business benefit, novalty, complexity, and CMM level of the development organisation, together with any regression testing due to a development time line involving multiple functional releases.



Requirements traceability and cross referencing to maintain self consistency

Figure 9 - Traceability from objective to product sets

Use cases can also call on common displays, business rules, algorithms, and interfaces. Each interface or display for example being used by a number of use cases. Good requirements practice will provide traceability between these elements and flag up inconsistencies or show the impact of changes or test failures. Understanding the impact at the business level also allows better decision making on allowing waivers or deviations when such situations arise.

We work through questions sets about functional failures (perhaps because an input was not working or erroneously working - not the software was not working), effects of external intrusion, or security or sabotage. When we find particularl scenarios we develop derived requirements to ensure the system or machine operates within acceptable boundaries, by specifying checking requirements, close down requirements, alarms or warnings. Domain knowledge must be captured along with interface standards and its future maintenence, e.g. direct passing or a procotol converter and data ownership and security requirements on the interfacing and adjacent systems, and recording where procedures or specialist training or operating rules will be needed.

Traceability from use case steps, pre and post conditions and guarantees can be added to not only the class or object but to the methods and attributes or database tables and similar the platforms and infrastructure, even variants of the infrastructure, the software runs on. This provides a total top to bottom tree and shows how all the elements fit together, from the organisational structure, the roles and organisational allocation, to the business processes, rules, locations, and the software and infrastructure that implements the processes.

Traceability is critical where the overall objective or business requirement is an emergent property of a collection of lower level requirements or is realised though a number of subsystems or products. An emergent requirement can not be allocated to a single product - it is the collective interaction which allows its realisable. Traceability allows this to be seen and progress to be monitored. Emergent properties require good management to ensure they are realised.

7.5 Verification, Validation and Test

<<develop wording>>

8. Interfaces to risk and issues management

Each requirement or domain knowledge item for example can have an issue or assumption associated with it. This translates to uncertainty and hence risk.

By provide an easily accessible collaboration method, for example publishing over a network, people can be encouraged to contribute issues and risk information related to each requirement, domain item, or decomposition logic or rational. This are then logged and reviewed and processed within the risk management framework. All issues and assumptions can be traced to a common module and can be summated and linked to the planning and scheduling process and each project or programme risk management activities.

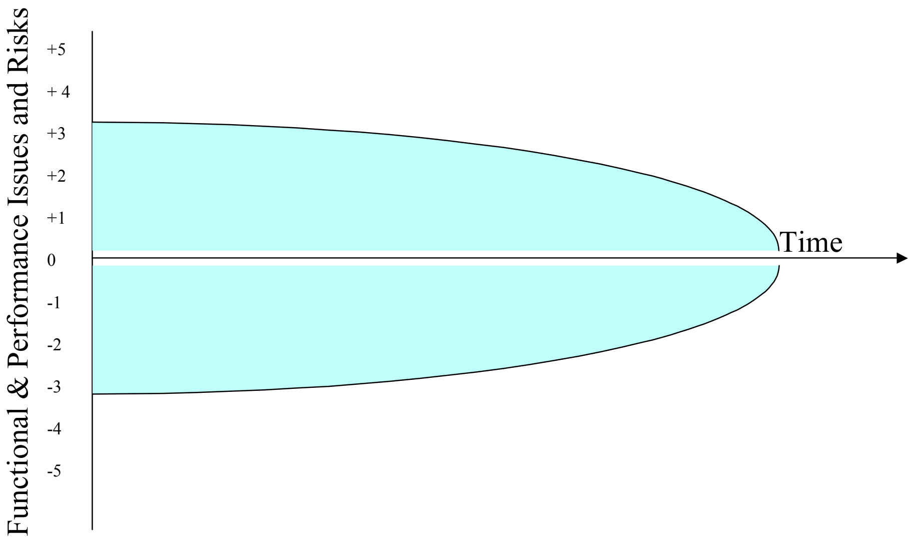


Figure 10 - Uncertainty & Risk

By linking the raised issues and assumptions to a defined project milestone, such as PDR, CDR, FDR, TRR or iteration point for example, changes in risk change be seen, drivers can be identified, and action can then planned for the next phase to reduce the assumption or risk by refining understanding or conducting a trial. Hence the planning process is fed with up to date information for use in next stage planning. This is fundamental to iterative based development as shown in Figure 12 - Change Management Flow ChartFigure 12. Over time the uncertainty can be seen to reduce primarily because it is being actively managed.

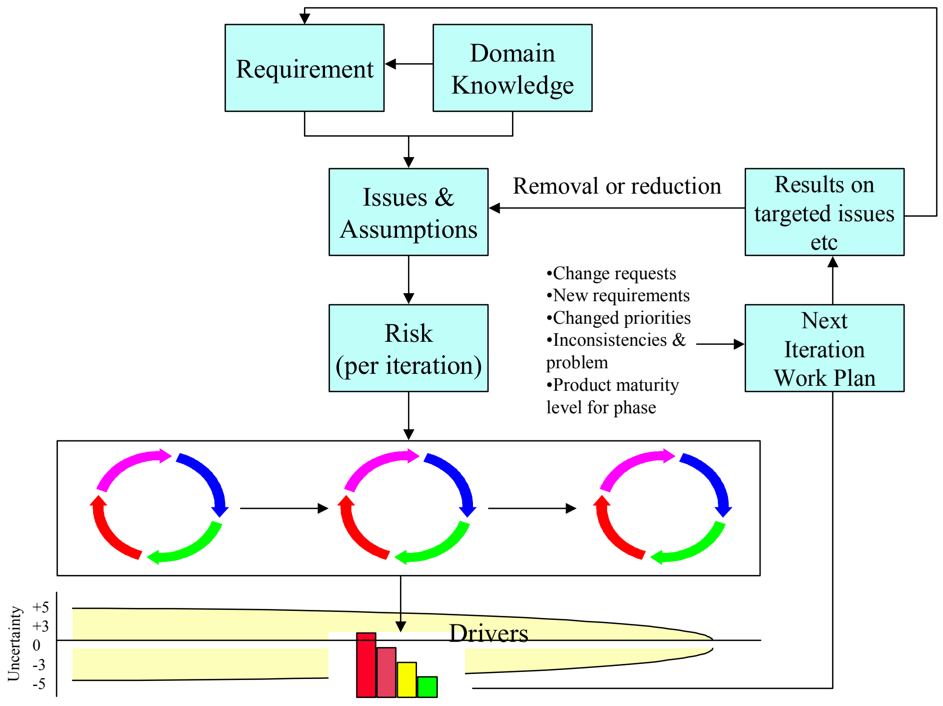


Figure 11 - Issue & Risk Management Process Linked to Requirements

9. Requirements, the customer and the organisation

Initial market analysis when developing a product or service and obtaining feedback from customers following release all forms part of the requirements management process. As we have seen various disparate groups in an organisation are involved in the requirements process. A good implement of the requirements management process uses an enterprise solution, with a central repository, which everyone can read, comment, and add to, and has the following benefit;

- better visibility of information, creating better awareness between users, and allowing information to be quickly spread, they email and log alerts if necessary.
- allows people to see the overall process, who uses information down stream of them, so allowing them to understand how it will be used, so they see those down stream as customers and have some structured liaison with those individuals

- minimises the rekey and structuring of information as it is passed between groups and suppliers, using a range of applications, normally ranging from word processors, spreadsheets, data bases, and drawing packages
- rekey also has risks as well as poor resource utlisation. The risk is that as rekey occurs in context and exact requirement become lost leading to specification errors.
- as the information exists across a number of documents configuration management becomes a real issue, as changes are proposed it is hard to see the overall effect, and when they are made to identify all impacted documents, change them, and track the information to give confidence that the documents have actually been updated.

Beaver Computer Consultants are experienced in using a number of requirements tools that can be used in networks and standalone machines sharing data by disks, using various application packages. Some are MS Word add-ins, togethers are databases with a simple interface. For those requiring only view access and the ability to add change requests or make comments or observations, an internet/intranet version is available, using a web browser and reads the central requirements database.

10. Change Management and Baseline Lines.

10.1 The basic change process

During the life of a project changes are to be expected. This is reflected in modern development processes. Therefore change control policies and a process will be needed. A typical basic process is shown in Figure 12 - Change Management Flow ChartFigure 12.

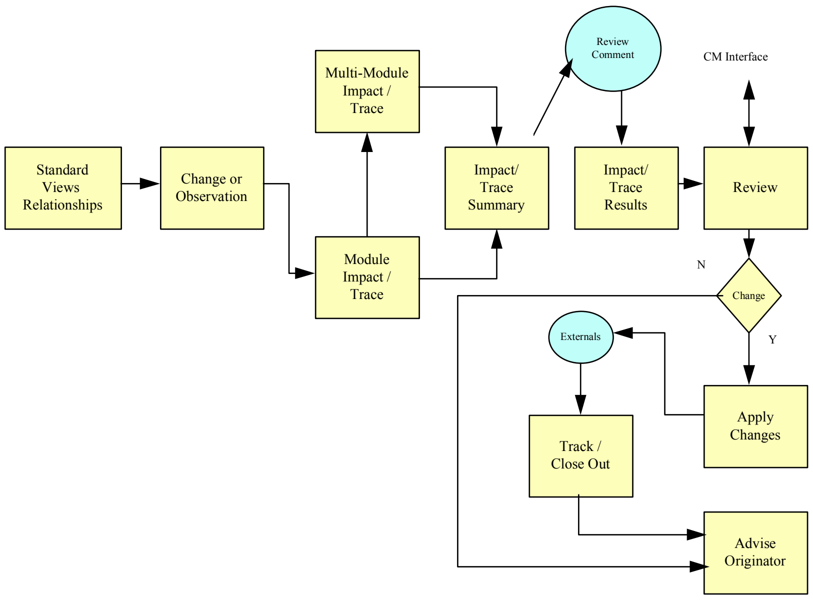


Figure 12 - Change Management Flow Chart

Essentially the process uses traceability matrices or links to identify from a proposed change to a requirement, its traceability to the higher level (they may be several), and ask whether the change is consistent and complete with that high level requirement, and using impact tracing to lower level requirements to see how they need to changed as well and to propose changes as necessary.

The set of proposed changes is document and the impact on usability, cost, performance, and other factors is evaluated by each of the specialist decisions, and the results examined by a change board. The decision is logged and either the changes tracked and progressed or the change is reject. In all cases the originator is kept informed.

A cautionary note is needed when examining batches of changes and the baseline used. A change may be examined against a baseline and approved, as might another. However if the second proposal had been made in light of the first proposal being implemented it might not have been independent. Similar issues occur where value engineering drives changes, as subsequent changes may not yield the savings of the baseline as reductions have already occurred - its a 10% saving of 90% of the baseline. Laws of diminishing returns can soon apply.

10.2 Baselines – maintaining sets of documents or data

The change management process also needs to consider the concept of creating baselines. A baseline is collection of related and uniquely versioned documents or data sources. Baselines can be thought of as a mechanism to synchronise (or least understand where they are not synchronised) information at defined points in time. Typically baselines will be made for;

- The User Requirements Set
- The System Requirement Set
- The Solution Set

The baseline of the Solution Set would reference a baseline for the System Requirement Set, which in turn would be. This approach provides fixed references for given points in time or milestones in the development life cycle.

For example the User Requirements could be at Version 3, System Requirements at Version 4, and so on. When published the baseline would include, perhaps in a release note, a summary of the consistency in terms of verification between the documents of data sources, the changes made against the previous baseline and so on. Baselining sets of documents or data sources becomes more important with large projects, particularly where a number of subsystem requirements and solution documents are developed.

Some projects may also maintain different baselines for what the requirements or a solution will be for fixed points in time. These are sometimes referred to as "blue-prints". For example a business may have requirements for the years 2003, 2005 and so on, each set related to its business processes, reflecting the business strategy and evolution. This means a number of solutions can exist, traced to the relevant business requirements. The change management process needs to consider a change not only against the applicable blueprint, but also subsequent blueprints, and if approved, reflect the change in each successive blueprint (requirements, solution, risks etc). Likewise if a defect is found (either in the requirements or solution) which drives a change, it must be updated for each applicable blueprint e.g. 2003, 2005. Sometimes when evaluating changes, say against 2003 it can be accommodated, but when considered for 2005 it may conflict, or need to be implemented by a different solution because the solution for 2005 takes account of other requirements at that time point.

10.3 Configuration management and publishing

When an individual document or date source is published, such as making it available for use by another work group, it needs to be versioned and archived. The same applies to a baseline set of documents or data sources. This publishing process is initiated by a need to publish, either in readiness for a review, or for use by another work group. Publishing will require making appropriate archives or versions of the document(s) or data source(s) and adding a note on the version / publishing history record.

A variation of this process is "quarantining" a document or data source to control, limited or stop changes to the artefact such as when it is being reviewed or tested. After that only changes arising from the review will be made. It then passes on to the next stage for final publishing, issuing any release note, placing into the configuration library, and issuing as appropriate using a controlled distribution process. The latter can be important in ensuring all authorised users or recipients receive updated versions, and removing from use previous versions.

When using tool support, particularly with links between different types of information, when a document is versioned or baselined consider if any links to other documents or data sources are used to create information included in the document. For example a solution document contains a main section and another section consisting of a trace matrix. When versioning consider if a version of the linked source also needs to be made, to ensure if an older solution document is recovered, the links or information in that document (or more accurately the tools data file) will point to the correct version of the data in the secondary file which is used to create the trace matrix. The link can still exist between the data items but the actual content of each can differ between versions. In some cases it is necessary to make version copies of each data file and the final published media, such as a Word document, and commit these to the configuration library along with the release note and the updated master version / baseline index.

11. Tools

While a tool is not going to automatically provide good management of requirements, they do simply the task for management complex volumes of data, and maintain change records. This is more effective that paper or spreadsheet methods, and easier than trying to develop your own RDDMS and most tools include various support for importing and exporting to word processors, spreadsheets, and DTP applications, and have ready to go tools to get you going and going quickly, for example impact and trace tools, change proposal and management tools, and filtering and search tools. Many applications, with varying features, and a wide cost range, a selection of tools is given in Table 9.

Table 9 - Requirements Tools

Tool	**Internet Address**	**Indicative Cost Band (basic) UK Sterling**			
		<500	<2000	<5000	<10000
Specialised Requirements Tools with process capability					
Calibre RM	www.borland.com			X	
DOORS	www.telelogic.com			X	
DOORS RequieIT	www.telelogic.com	X			
DOORSNet / 10 users	www.telelogic.com				X
Rational Requite Pro	www.rational.com		X		
Requirements Analyst	www.analysttool.com	X			
Team Trace	www.teamtrace.com	X			
RTM	www.chipware.com			X	
Integrated Process Tools with requirements functionality					
Focal Point	www.focalpoint.se			X	
Speedev	www.speedev.com		X	X	

Interated systems engineering tools with requirements functionality					
RDD	www.holagent.com			X	X
CORE	www.vtcorp.com			X	X
Cradle	www.threesl.com			X	X

A sumary of features and comparisons from a requirements tools survey is available at the INCOSE web site (www.incose.org). Beaver Computer Consultants are DOORS users.

Unfortunately most tools do not contain a process framework, instead you'll need to structure this yourself: creating document modules, setting up attributes, and creating views for various users, relevant to your defined process. This paper will hopefully give you sufficient information to make a start on this. Start off by focusing on embedding good practices and on finding solutions to existing process or implementation problems you currently have. Alternatively Beaver Computer Consultant can help you to do this, mentoring and training your team, while working with you.

It is important to ensure you maintain referential integrity between attributes. For example a requirement might have attributes for rationale, the review status, and test status. If the rationale changes then review status and test status will need reviewing again and potentially updating. Integrity can be maintained in a number of tools by using custom scripts other tools provide a process flow framework to simplify this task.

12. Implementing a process

Having set up an information model for the requirements and associated documents, and defined attributes that are meaningful and customised for the company, you may decide to provide tool support. As a minimum a Requirements Process Manual should be introduced, detailing the process, with supporting guidance, and worked examples. Provide initial training and mentoring. This is often best achieved using constultancy support for a short period. Beaver Computer Consultants have under taken a number of such assignments.

If using a software requirements application remember that while they provide features in a box they generally need customising. This also ensures a standard approach is used in the company. Aim to guide users though using the tool, provide a customise user menu interface, which implements the process in sequence and prompts with guidance and tool support. This reducing the skill level in using the software and leaves users to get on with the job.

As you have seen various attributes have criteria or guidance associated with them. These can be implemented by apply multiple item lists or enumerated lists to the attributes. Think about defining a set of process views, which apply to a given user or class of user, at a point in the process, which show just the requirement, the relevant attributes, or requirements that match an attribute, with field for recording new information that is required to be put in. You can also provide logic in a tool to check compliance with the process, whether links have been established in an appropriate way, and to high light suspect information. In additional consider;

- Setting users, access and security levels
- Views that match the process
- Providing import/export tools, and tools to assist in parts of the process such as 2x2 matrix generators
- On line summary reports and a dash board which showing the level of process, and suspect links or attributes

A further feature of tool support is that they provide a means to publish requirements and information to a wide audience, quickly. This can be used to get people involved, particularly if commentary and change proposal features are implemented so people can flag up information, issues, and suggest additional references or data. The tool provides an enabling mechanism to get people working together collectively and collaboratively focusing on common objectives.

13. Summary : Requirements engineering implementation

The following list summarises the activities discussed in this paper between 2 levels in a supply chain.

- Agree the overall process covering collecting stakeholder needs, requirements formulation, quality tests and reviews, supplier negotiation, change control, and compliance monitoring.
- Agree a process for elicitation, analysis, testing, release, design review, and V&V planning, evidence collecting and acceptance.
- Outline and agreed a format and contents of a requirements specification and agree how it will be maintained
- Define an information model, that is the types of specifications, test documents, risk registers, domain knowledge, issues log, glossary, and the defined types of links between them. You could implement your model using a word processor or spreadsheets, in which case you'll also need some trace matrix between them, unless you use hyperlinks or HMTL. Alternative you might develop your own RDBMS or use a "out of the box" requirements tool of the type outlined in Section 11. Also consider the internal structure of key specifications such as the user requirements and system requirements specification. Good structuring will promote better traceability and if tools are to be used, allow features of the tools to be used. Pay attention to the level of detail to be held in each requirement and how this will related to validation and compliance monitoring statements.

- Plan and implement tasks to populate the requirements, initially input or import existing stakeholder needs or proposed requirements, and assign relevant attributes.
- Where necessary import existing requirements, evaluate or test them, and restructure. Formulate each requirement, trawling for information to compose it, including identifying domain information, assigning additional attributes.
- Test the formulated the requirements for goodness of fit completing the test quality attributes.
- Check the requirements meet the high level requirement and orphans at either level no don't exists, completing further test quality attributes.
- Conduct a requirements review and release the requirements specification to the level in the process or supplier.
- The next process team or supplier review the flowed down requirements and negotiate back with the higher level team.
- Once accepted the lower level team plan V&V / tests against each requirement for each stage and submit back to the higher level for approval.
- Simultaneously they start to decompose the requirements, producing either a lower level specification or develop a design solution. This includes identifying candidate products and technologies, the functional allocation to the infrastructure, applications, and other equipment, and people, and conducting sufficient analysis, simulation, and refined prototyping to ensure the high level requirements are being achieved. Appropriate requirement types, attributes and links are set.
- The lower level team collect design review evidence and ensure each requirement is addressed. A requirements or design review meeting is held.
- At key stages in the project life cycle V&V evidence is collected and review, and submitted to the higher level team for acceptance for the particular stage. This indicates that the solution is being realised and the level of risk and uncertainty is reducing.
- Manage changes, trace the impact of proposed changes, establish the effect, and determine whether to accept or reject, and trace and update all affected requirements, documents, and tests as traced.

Through out the process, regularly management reports need to be produced, detailing the level of produced made through the process, that is how many requirements have progressed to each level, and if they have been accepted or not.

14. Beavers Services

We hoped this paper has proved useful to you. If you need assistance in developing a specification or testing whether a design meets a specification Beaver have a range of services that can help you, including;

- ***Requirements Elicitation*** using document studies, use case and scenario work shops, and existing system studies, and subsequent ***Analysis*** involving categorisation, structuring, adding attribute and traceability information, and ***Specification Development***.

- ***Trawling and decomposition of requirements*** including applying various forms of analysis or extracting information from existing analysis reports or product documentation.
- ***Process definition, training, and mentoring of staff***
- ***Building information models*** *– setting up views, modules and documents to ensure a high integrity hierarchy of specifications is developed.*
- ***Importing existing documents and legacy data into a requirements tool***
- ***Setting up*** requirements management tools, ***configuring*** and ***customising toolsets*** to support a defined process, and support and training staff and suppliers.
- Support for ***ITT and tender development*** and ***bid - proposal response*** and ***tender assessment***
- Design and ***solutions development*** and ***mapping back to*** system and test / V&V ***requirements***

Beavers unique proposition is its experience in requirements management, and the supplementary experience in analysis, modelling, baselining, performance studies, to populate a specification or V&V a specification against the specification up at the next level, operated within a structured PRINCE based project management framework. We promote collaborative team working.

15. References

Author	**Title**	**Publisher**	**ISBN**
Capers Jones	Patterns of software systems failure and success	Thomson Computer Press	1-850-32804-8
Dick, J	Summer 2000 edition of QSS News Bytes.	QSS Inc. Now Telelogic.	NA
Jackson, M	Software Requirements and Specifications : a lexicon of practice, principles and prejudices	Addison-Wesley	0-201-87712-0
Standish	CHAOS Report	Standish Group	NA
Sommerville, I & Sawyer, P	Requirements Engineering : A Good Practise Guide	Wiley	0-471-97444-7
Robertson, S & Robertson, R	Mastering the Requirements Process	Addison-Wesley	0-201-36046-2
NA	PSS 05. Software Requirements Procedures	European Space Agency	NA
Macaulay,L	Requirements	Springer	3-540-76006-7

	Engineering		
Telelogic / QSS	Getting it right first time : writing better requirements.	QSS / Telelogic	NA