7 Advanced Traceability

For starters I'll have "Who?", "What?", "When?", "Where?", and then "Wither?", "Whence?" and "Wherefore?" to follow, and one big side order of "Why?"
Zaphod Beeblebrox in *The Hitch-Hiker's Guide to the Galaxy*
Douglas Noel Adams, writer, 1952–2001

7.1 Introduction

So often, the real rationale for a particular design and the deeper understanding of how the components of a system work together to achieve an end result remain in the minds of the engineers. Months or years later, when the original designers have long since moved on, or their memory has dimmed, the loss of that understanding may seriously impede the ability to evolve, maintain or reuse the system.

This chapter first presents a technique for maintaining this greater understanding of a system, through capturing the rationale associated with the relationships between problem, solution and design. Christened "rich traceability", the approach builds on the basic concepts of "elementary traceability" presented in Chapter 1 and applied in subsequent chapters.

While rich traceability may represent one big side order of "Why?", the "Wither?", "Whence?" and "Wherefore?" of traceability are perhaps addressed through another subject of this chapter: metrics in relation to traceability.

7.2 Elementary Traceability

There are many ways of representing many-to-many relationships. One consultant visited a defence contractor just prior to a customer traceability audit to find the office all laid out ready. Along the length of the floor on one side was spread out the requirements document and on the other side the code listing. Traceability was shown by pieces of string taped between the documents. Space consuming, time consuming, non-maintainable and non-transportable. But it did some of the job.

Many engineers will have seen traceability represented in matrix form as an appendix to relevant documents. The two dimensions identify, for instance, user requirements on one axis and system requirements on the other, with marks in those cells where a relationship exists.

There are a number of disadvantages to this approach:

- Where there are a large number of statements to index on both axes, the paper or screen is too small to show enough information.
- Traceability relationships tend to be sparse, resulting in most of the cells in the matrix being empty, which is a waste of space.
- It is very hard working your way through multiple layers of traceability presented in a number of separate matrices.
- Information about traceability is separated from the details of the requirements themselves.

Another method is to use hyper-linked documents, where statements can be highlighted, linked to other statements and traversed at will – in either direction if you are clever. Now the traceability information is visible in the text of the statement, but there are still problems:

- To carry out analysis you may have physically to traverse the link before text at the other end is visible.
- It is hard to spot when the item at the other end of a hyperlink has been deleted, leaving a dangling link, making traceability difficult to maintain.

Whatever approach you use, unless supported by a tool, traceability will be very hard to manage.

The simplest form of traceability is achieved by linking statements together using some kind of database support. It is helpful if linking information is held separately from the documents. It is essential that statements are independently and uniquely identifiable.

With analysis in mind, the essential capabilities for implementation of traceability are:

- ability to create links between statements, thus forming permitted relationships;
- ability to delete links between statements in a controlled manner;
- ability to view simultaneously the text (or other attributes) of statements at both ends of a selected relationship;
- ability to carry out coverage analysis to show those statements covered or not covered by a selected relationship;
- ability to carry out single- and multi-level impact analysis to show sets of impacted statements;
- ability to carry out single-level and multi-level derivation analysis to show sets of originating statements;
- ability to carry out upwards and downwards coverage analysis to show sets of statements covered and not covered by selected relationships.

Figure 7.1 shows an example of elementary traceability. A user requirement traces down to three responding system requirements. In this presentation, the text of the user requirement is visible together with the set of system requirements that respond to it. Having this information together allows the traceability to be reviewed easily. Figure 7.2 shows a second example.

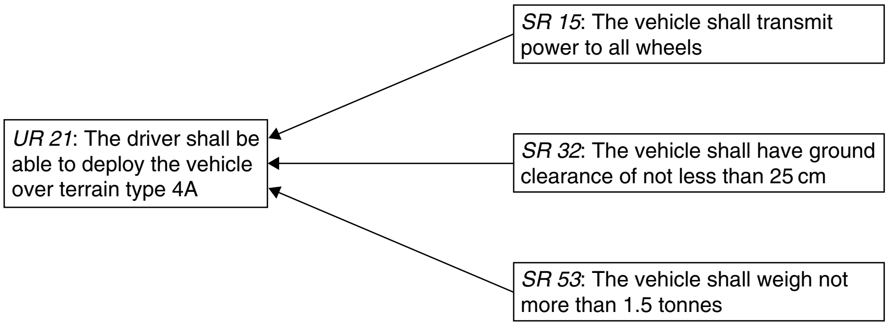

SR 15: The vehicle shall transmit power to all wheels
UR 21: The driver shall be able to deploy the vehicle over terrain type 4A
SR 32: The vehicle shall have ground clearance of not less than 25 cm
SR 53: The vehicle shall weigh not more than 1.5 tonnes

Figure 7.1 Elementary traceability example: military vehicle.

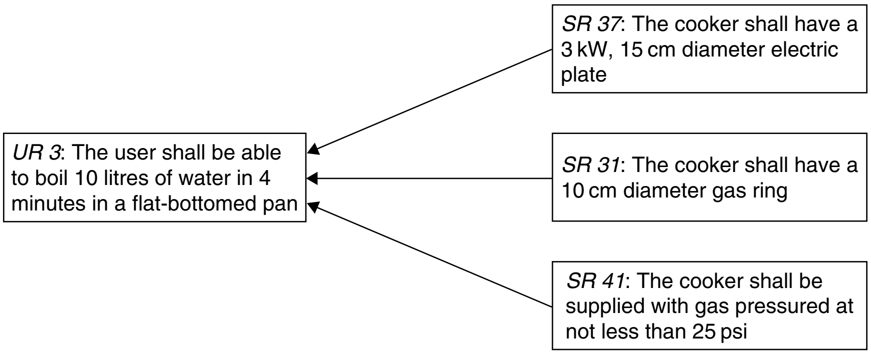

SR 37: The cooker shall have a 3 kW, 15 cm diameter electric plate
UR 3: The user shall be able to boil 10 litres of water in 4 minutes in a flat-bottomed pan
SR 31: The cooker shall have a 10 cm diameter gas ring
SR 41: The cooker shall be supplied with gas pressured at not less than 25 psi

Figure 7.2 Elementary traceability example: cooker.

7.3 Satisfaction Arguments

Implementation of elementary traceability as discussed in Section 7.2 represents a major step forward for many organizations. Indeed, changing the culture of an organization to embrace even this simple approach may be a big enough leap in itself. However, there is, as always, more that can be done.

The intention in the example of Figure 7.1 is that the three system requirements are somehow sufficient to satisfy the user requirement. It is difficult, however, for a non-expert to assess the validity of this assertion. This is because the reasoning has not been presented.

What is better is to present a "satisfaction argument" for each user requirement. With the elementary traceability of Figure 7.1, the only information provided is that the three system requirements play some kind of role in the satisfaction argument, but there is nothing to indicate exactly what the argument is.

Rich traceability is a way of capturing the satisfaction argument. This appears as another statement sitting between the user requirement and the corresponding system requirements, as illustrated in Figure 7.3.

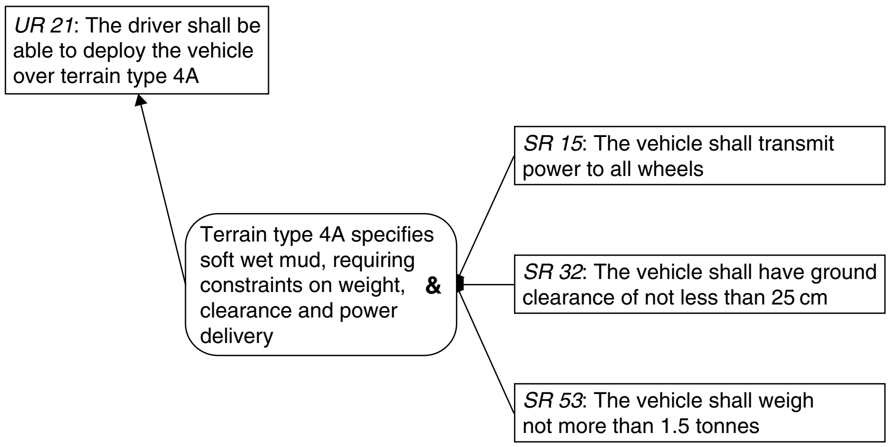

UR 21: The driver shall be able to deploy the vehicle over terrain type 4A
Terrain type 4A specifies soft wet mud, requiring constraints on weight, clearance and power delivery
&
SR 15: The vehicle shall transmit power to all wheels
SR 32: The vehicle shall have ground clearance of not less than 25 cm
SR 53: The vehicle shall weigh not more than 1.5 tonnes

Figure 7.3 Rich traceability example: vehicle.

Not only is the satisfaction argument expressed textually, but an indication is given about the way in which the system requirements combine in the argument using a propositional operator:

- by conjunction (**&**), indicating that the contribution of *all* the system requirements is necessary for the user requirement satisfaction argument to hold;
- by disjunction (**or**), indicating that the contribution of *any one of* the system requirements is necessary for the user requirement satisfaction argument to hold.

An example of disjunction is given in Figure 7.4, where satisfaction is achieved through provision of *either* an electric ring *or* a gas ring *or both*. Note the two-level propositional structure of the argument.

Much more information is now provided about how the user requirements are being satisfied. Even one who is not a domain expert may feel capable of assessing important aspects of the argument. The text helps in assessing the logic of the argument for validity and completeness. The operator makes the structure of the argument more precise.

Notice in particular that it is not at all clear in Figure 7.2 that the set of system requirements represent alternative solutions, whereas in Figure 7.4 the fact is made absolutely specific. If an electric ring cannot be supplied, the requirement can still be satisfied through a gas ring.

The authors first came across the concept of rich traceability in the Network Rail (then Railtrack) West Coast Route Modernization project in the UK, where a team from Praxis Critical Systems had devised a requirements management process and data model that used "design justifications". The same concept can be identified in a variety of similar approaches in which satisfaction arguments are called variously "requirements elaboration", "traceability rationale", "strategy", etc.

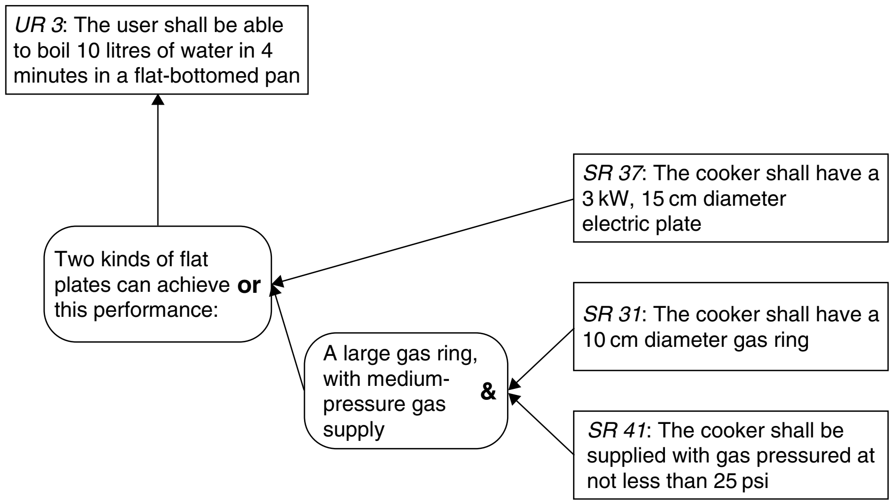

UR 3: The user shall be able to boil 10 litres of water in 4 minutes in a flat-bottomed pan
Two kinds of flat plates can achieve this performance: **or**
SR 37: The cooker shall have a 3 kW, 15 cm diameter electric plate
A large gas ring, with medium-pressure gas supply **&**
SR 31: The cooker shall have a 10 cm diameter gas ring
SR 41: The cooker shall be supplied with gas pressured at not less than 25 psi

Figure 7.4 Rich traceability example: cooker.

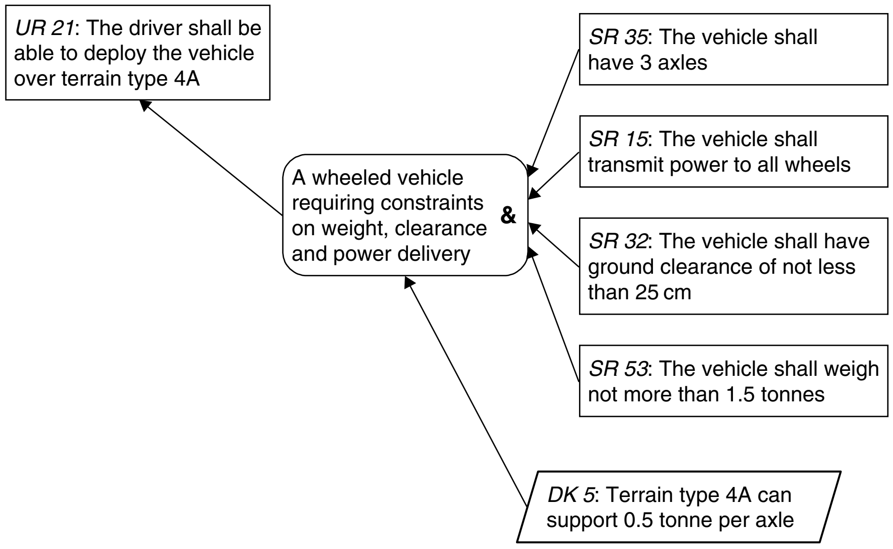

UR 21: The driver shall be able to deploy the vehicle over terrain type 4A
A wheeled vehicle requiring constraints on weight, clearance and power delivery **&**
SR 35: The vehicle shall have 3 axles
SR 15: The vehicle shall transmit power to all wheels
SR 32: The vehicle shall have ground clearance of not less than 25 cm
SR 53: The vehicle shall weigh not more than 1.5 tonnes
DK 5: Terrain type 4A can support 0.5 tonne per axle

Figure 7.5 The role of domain knowledge.

Satisfaction arguments may depend for their validity on things other than lower level requirements. Figure 7.5 shows an example using "domain knowledge" to support the argument. Domain knowledge is a fact or assumption about the real world and not something that constrains the solution in and of itself. In this case, the statement of domain knowledge is an essential part of the satisfaction argument, shown in a slanted box.

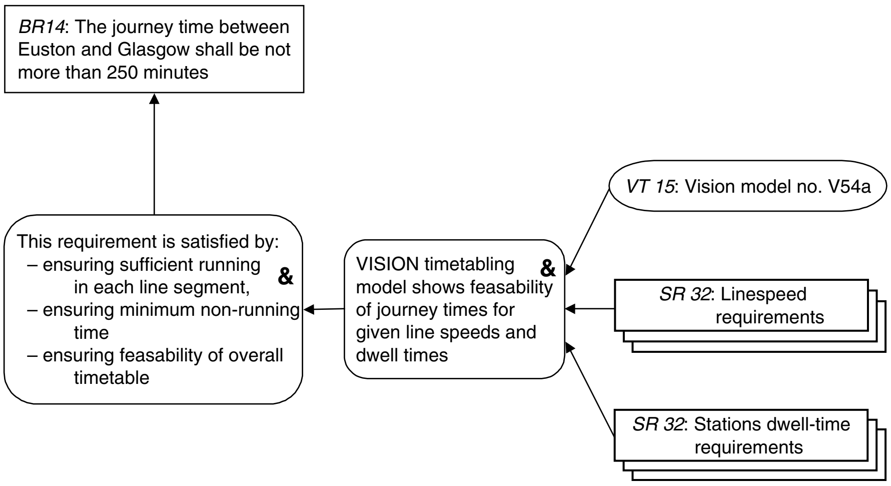

BR14: The journey time between Euston and Glasgow shall be not more than 250 minutes
This requirement is satisfied by:
– ensuring sufficient running in each line segment,
– ensuring minimum non-running time
– ensuring feasability of overall timetable
&
VISION timetabling model shows feasability of journey times for given line speeds and dwell times
&
VT 15: Vision model no. V54a
SR 32: Linespeed requirements
SR 32: Stations dwell-time requirements

Figure 7.6 The role of modelling.

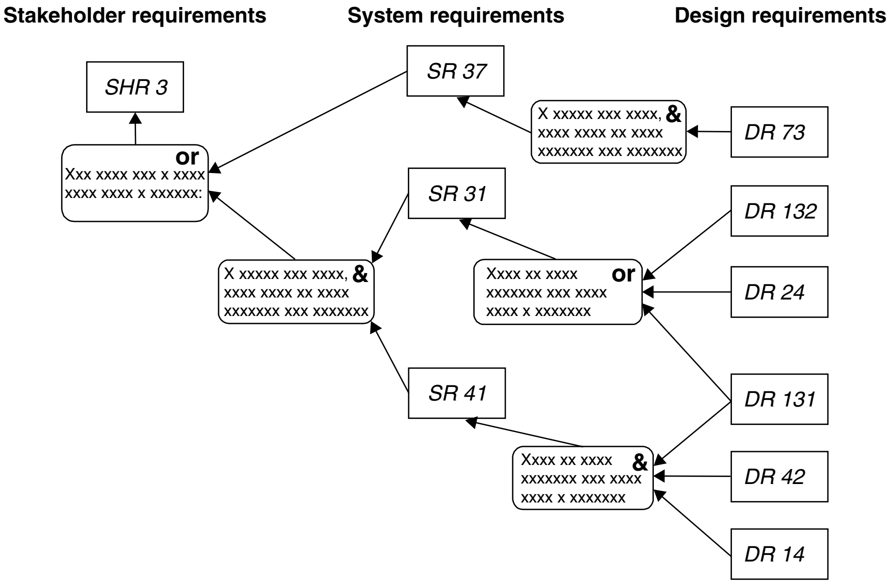

Stakeholder requirements
System requirements
Design requirements
SHR 3
or
Xxx xxxx xxx x xxxx xxxx xxxx x xxxxxx:
SR 37
X xxxxx xxx xxxx, xxxx xxxx xx xxxx xxxxxxx xxx xxxxxxx
&
DR 73
SR 31
DR 132
X xxxxx xxx xxxx, xxxx xxxx xx xxxx xxxxxxx xxx xxxxxxx
&
Xxxx xx xxxx xxxxxxx xxx xxxx xxxx x xxxxxxx
or
DR 24
SR 41
DR 131
Xxxx xx xxxx xxxxxxx xxx xxxx xxxx x xxxxxxx
&
DR 42
DR 14

Figure 7.7 Multiple layers of rich traceability.

Capturing such assumptions is important, not least because the world, and the assumptions one can make about it, have a habit of changing. Once captured, derivation analysis can be used to understand the impact of changing assumptions on the ability of the system to meet its requirements.

An example of this comes from the New York underground. A series of accidents in the 1970s were due to a false assumption concerning the stopping distance of trains. Initially valid, the assumption was invalidated as trains got heavier over the years, and the stopping distance increased. Although the performance of the signalling software was originally correct, and it did not evolve, the changing assumptions meant that it ceased to meet requirements from a certain time.

The ability to document and trace the role of such assumptions is possible through effective traceability.

Another example of non-requirements information playing a role in satisfaction arguments comes from modelling activities. Satisfaction arguments are often derived from complex modelling activities, the complete details of which are too detailed to be captured in rich traceability.

Figure 7.6 shows an example abstracted from a railway project in which a satisfaction argument depends on the results of a complex timetable modelling activity using specialized software. A set of assumptions and subsystem requirements are derived from the modelling tool and these are documented in the rich traceability structure. The modelling reference is shown in a box with rounded ends.

In this case, the modelling activities that need revisiting become apparent under impact analysis.

Rich traceability can, of course, be used through multiple layers of requirements or objectives. Figure 7.7 depicts three layers and the traceability between them.

7.4 Requirements Allocation

The satisfaction argument is often trivial, amounting perhaps only to the allocation of an identical requirement to one or more subsystems or components. This is sometimes referred to as requirements “allocation” or “flow-down”.

Where this pure flow-down of requirements is used, the change process may be simplified. Changes to high-level requirements may be automatically flowed-down to lower levels.

A simple extension of rich traceability allows such cases to be captured. A new value representing “identity” is added to the “and” and “or” operators used to annotate the arguments. Figure 7.8 shows an example of this. The symbol “=” is used to indicate identity.

7.5 Reviewing Traceability

Every time a requirement is reviewed, it should be reviewed along with its satisfaction argument. Based on rich traceability, a review process can be established that focuses on one requirement at a time, together with its satisfaction argument, and the requirements that flow from it.

Figure 7.9 shows a screen shot of a tool used in a defence project to review requirements and satisfaction arguments. On the screen is just the right parcel of information to assess a requirement and how it is satisfied.

The dark triangles are for navigating downwards through the layers of traceability or across to the next requirement at the same level.

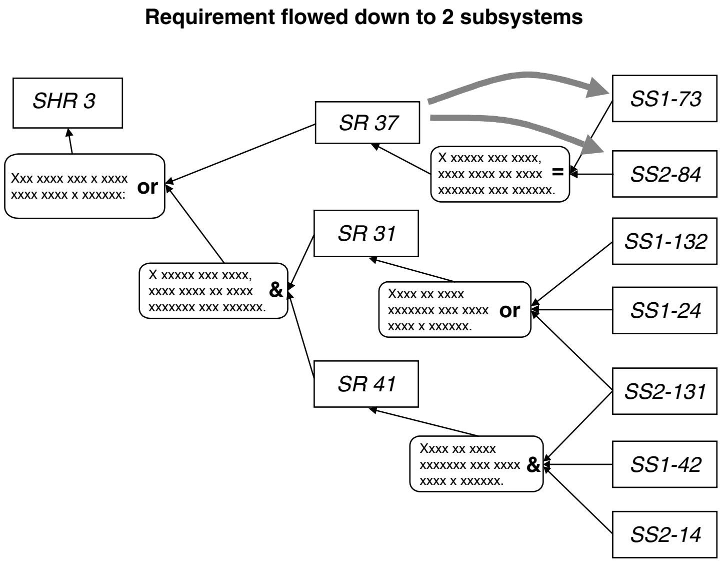
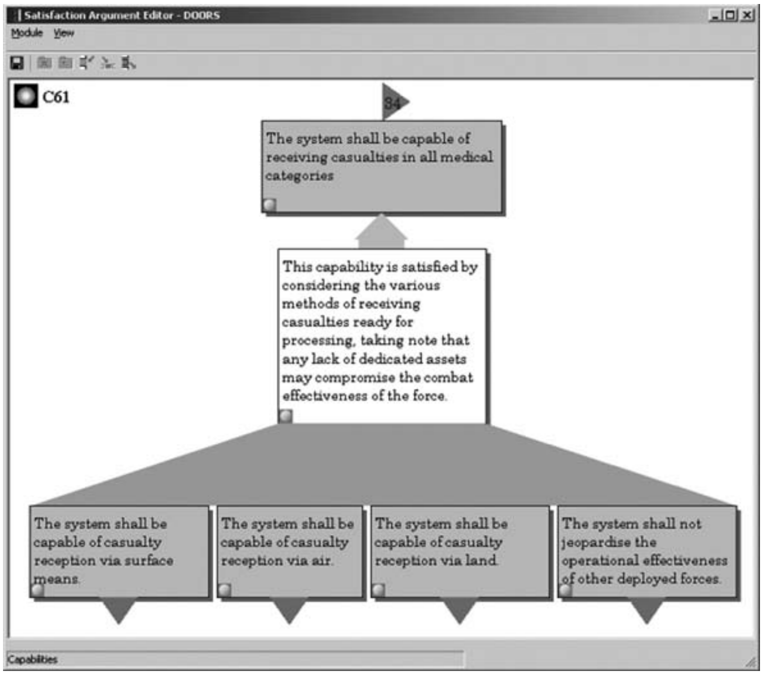

Requirement flowed down to 2 subsystems
SHR 3
Xxx xxxx xxx x xxxx xxxx xxxx x xxxxxx: or
SR 37
X xxxxx xxx xxxx, xxxx xxxx xx xxxx xxxxxxx xxx xxxxxx. =
SS1-73
SS2-84
SR 31
X xxxxx xxx xxxx, xxxx xxxx xx xxxx xxxxxxx xxx xxxxxx. &
Xxxx xx xxxx xxxxxxx xxx xxxx xxxx x xxxxxx. or
SS1-132
SS1-24
SR 41
SS2-131
Xxxx xx xxxx xxxxxxx xxx xxxx xxxx x xxxxxx. &
SS1-42
SS2-14

Figure 7.8 Flow-down of requirements using "identity".



The system shall be capable of receiving casualties in all medical categories

This capability is satisfied by considering the various methods of receiving casualties ready for processing, taking note that any lack of dedicated assets may compromise the combat effectiveness of the force.

The system shall be capable of casualty reception via surface means.

The system shall be capable of casualty reception via air.

The system shall be capable of casualty reception via land.

The system shall not jeopardise the operational effectiveness of other deployed forces.

Figure 7.9 Reviewing tool for satisfaction arguments.

7.6 The Language of Satisfaction Arguments

As with requirements, it helps to have a uniform approach to expressing satisfaction arguments. The key guideline is to start the sentence with "This requirement will be satisfied by …", which focuses the mind on the kind of statement being made.

While requirements should be strictly atomic (see Chapter 4), satisfaction arguments need not be so limited. However, if statements become too complex, a structured argument should be used instead.

Repeated patterns of satisfaction arguments may be identifiable, in which case a palette of boilerplate statements could be used to good effect.

7.7 Rich Traceability Analysis

The presence of satisfaction arguments in rich traceability does not preclude the ability to carry out elementary impact and derivation analysis as described in Chapter 1. Indeed, the arguments add important clues as to the nature of the impact by capturing understanding, or *raison d'être*.

The propositional structure (ands and ors) of the satisfaction arguments offers opportunities for other kinds of analysis. For instance, the structures can be analyzed to show the number of degrees of freedom that exist for meeting a particular objective.

Take the example of Figure 7.4. The proposition structure for *UR3* can be captured in the expression SR37 *or (SR31 and SR41)*. Using the laws of propositional logic, this can be converted to a special disjunctive form in which each disjunct shows one way of meeting the requirement:

[SR37 and (not SR31) and (not SR41)]
or [SR37 and SR31 and (not SR41)]
or [SR37 and (not SR31) and SR41]
or [SR37 and SR31 and SR41]
or [(not SR37) and SR31 and SR41]

In simple cases, this analysis may not seem that useful, but imagine more complex scenarios where there are hundreds of requirements in several layers with complex interactions. One may want to know whether there is *any* way of meeting the requirements and, if there is no way, then what the cause is – where the conflict exists.

7.8 Rich Traceability for Qualification

Rich traceability can be used in any traceability relationship. The discussion so far has been based on the satisfaction relationship, but it is also applicable to qualification. In this case, the "satisfaction argument" may be referred to as the "qualification argument" or "qualification rationale". All the same advantages of using satisfaction arguments apply to the qualification strategy.

7.9 Implementing Rich Traceability

We describe here two approaches to the implementation of rich traceability: single-layer and multi-layer.

7.9.1 Single-layer Rich Traceability

In this approach, illustrated in Figure 7.10, each high-level requirement has a single statement of satisfaction or strategy as an attribute, and multiple low-level requirements may flow from it in a many-to-many satisfaction relationship. Another attribute (not shown in the diagram) is used to type the argument as either a conjunction or a disjunction.

7.9.2 Multi-layer Rich Traceability

Here satisfaction arguments can be structured into multiple layers: a main argument attached (as an attribute or linked in an "establishes" relationship) to the requirement to be established, and a hierarchy of subarguments hang off of the main argument. Low-level requirements are linked to the subarguments in a "contributes to" relationship. This is shown in Figure 7.11.

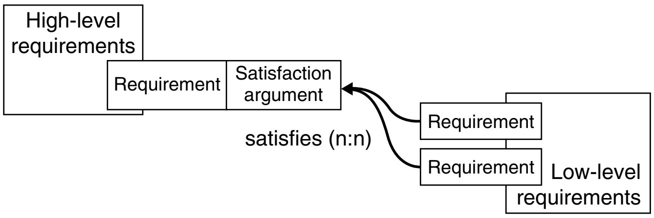


Figure 7.10 Single-layer rich traceability.

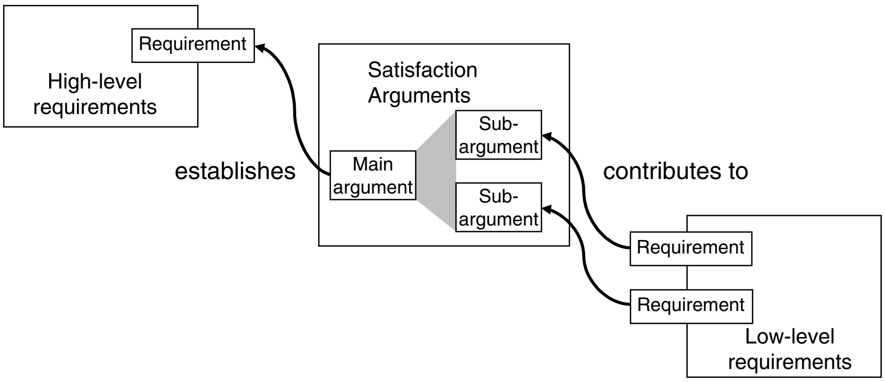


Figure 7.11 Multi-layer rich traceability.

Some implementations limit the depth of the argument hierarchy to two, using a main argument – the satisfaction argument – and a single layer of subarguments that explains the role played by the contributing requirements.

7.10 Design Documents

Astute readers will have noticed that the layer of rationale introduced by satisfaction arguments is very like the "filling" in the systems engineering sandwich presented in Figure 1.9. Indeed, the satisfaction arguments can be gathered into a document which may be best characterized as an "analysis and design" document. It is this design document which is the focal point of the integration between requirements and modelling. The role of the design document is to summarize – textually and visually – those parts of the modelling activity that explain why one layer of requirements is sufficient and necessary to satisfy the layer above. The document references data from the modelling process as evidence for the rationale. Traceability between layers of requirements passes through the design document. In this way, the results of modelling appear in the traceability chain and can engage in impact analysis.

Figure 7.12 portrays this approach. Layers of requirements are filled by design documents appropriate to the level of abstraction. The modelling activities at each level give rise to data that is referenced by the design document. The thin arrows represent the flow of information; the thick arrows represent traceability.

We now show an example of the kind of information that may be collected into a design document. A sequence of figures shows extracts from an "analysis of need" document that models a baggage check-in system at the problem domain level. The model sits between the "statement of need" and the "stakeholder

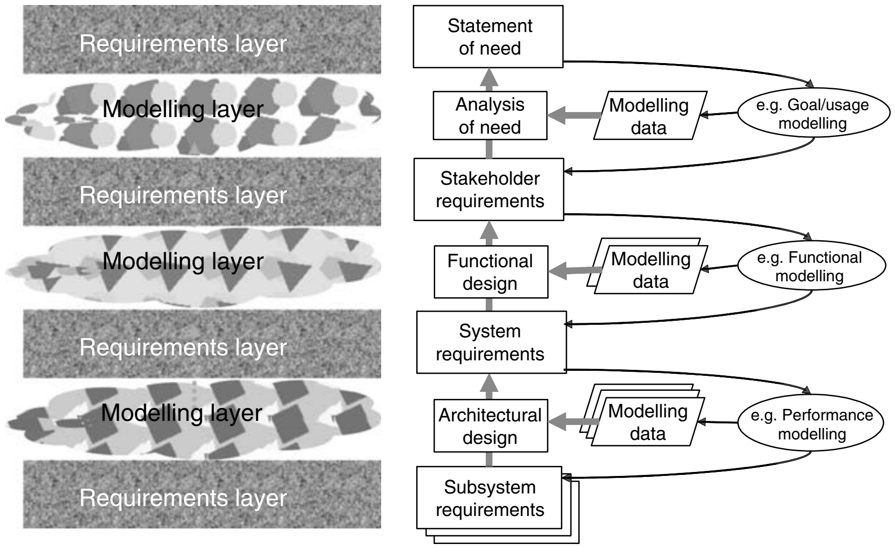
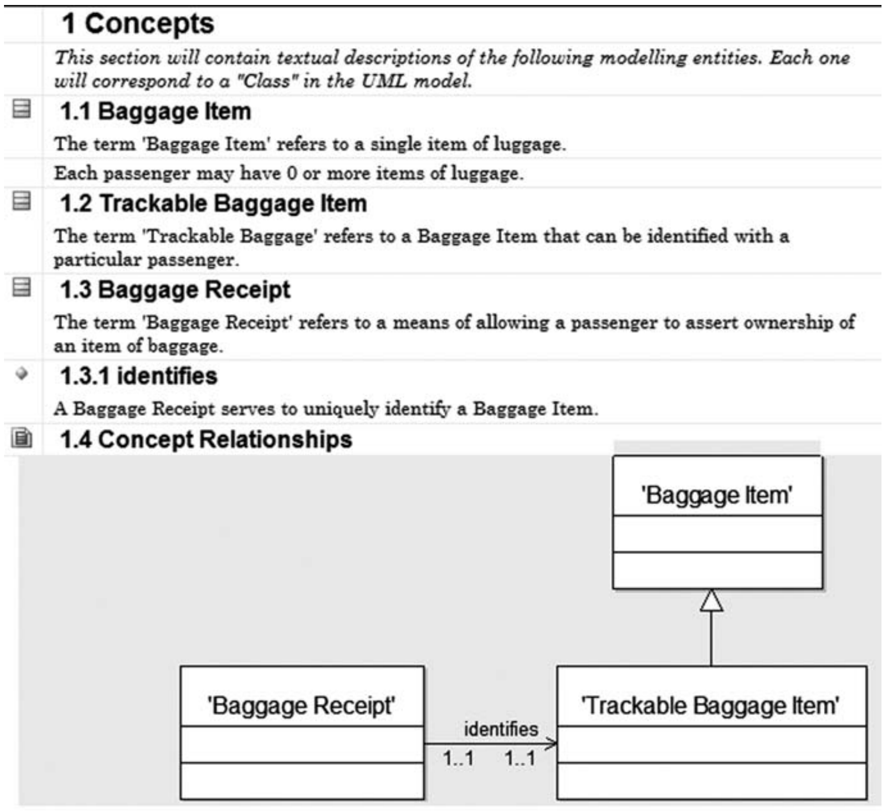


Figure 7.12 Analysis and design documents.



1 Concepts

This section will contain textual descriptions of the following modelling entities. Each one will correspond to a "Class" in the UML model.

1.1 Baggage Item

The term 'Baggage Item' refers to a single item of luggage.

Each passenger may have 0 or more items of luggage.

1.2 Trackable Baggage Item

The term 'Trackable Baggage' refers to a Baggage Item that can be identified with a particular passenger.

1.3 Baggage Receipt

The term 'Baggage Receipt' refers to a means of allowing a passenger to assert ownership of an item of baggage.

1.3.1 identifies

A Baggage Receipt serves to uniquely identify a Baggage Item.

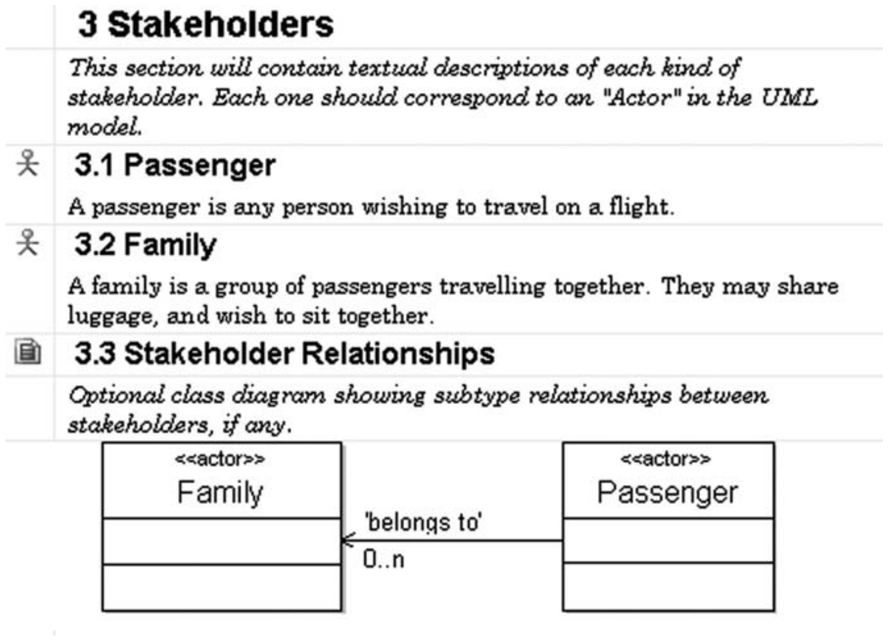
1.4 Concept Relationships

Figure 7.13 Concepts section of design document.

requirements" documents, and uses UML2 to portray the analysis in visual form.

The following kinds of information are typical:

- *Concepts.* A UML class diagram is used to identify the domain concepts and the relationships between them. Each concept is a UML class and each relationship is a UML association. Both appear as entries in the design document where a textual description of the concept or relationship is supplied. Figure 7.13 shows an example for the baggage check-in system. The symbols to the left of each paragraph indicate that that part of the document corresponds to a UML entity in the model.
- *Stakeholders.* This section lists the stakeholders that have been identified during analysis, and includes a class diagram showing relationships. In the example shown in Figure 7.14, there are two stakeholders with a single relationship.
- *Static context.* The purpose of this section (Figure 7.15) is to identify the context in which the baggage check-in system exists. The baggage check-in system itself is modelled as a class in a class diagram, along with classes representing all the surrounding and enclosing systems. Relationships between these systems



3 Stakeholders

This section will contain textual descriptions of each kind of stakeholder. Each one should correspond to an "Actor" in the UML model.

3.1 Passenger

A passenger is any person wishing to travel on a flight.

3.2 Family

A family is a group of passengers travelling together. They may share luggage, and wish to sit together.

3.3 Stakeholder Relationships

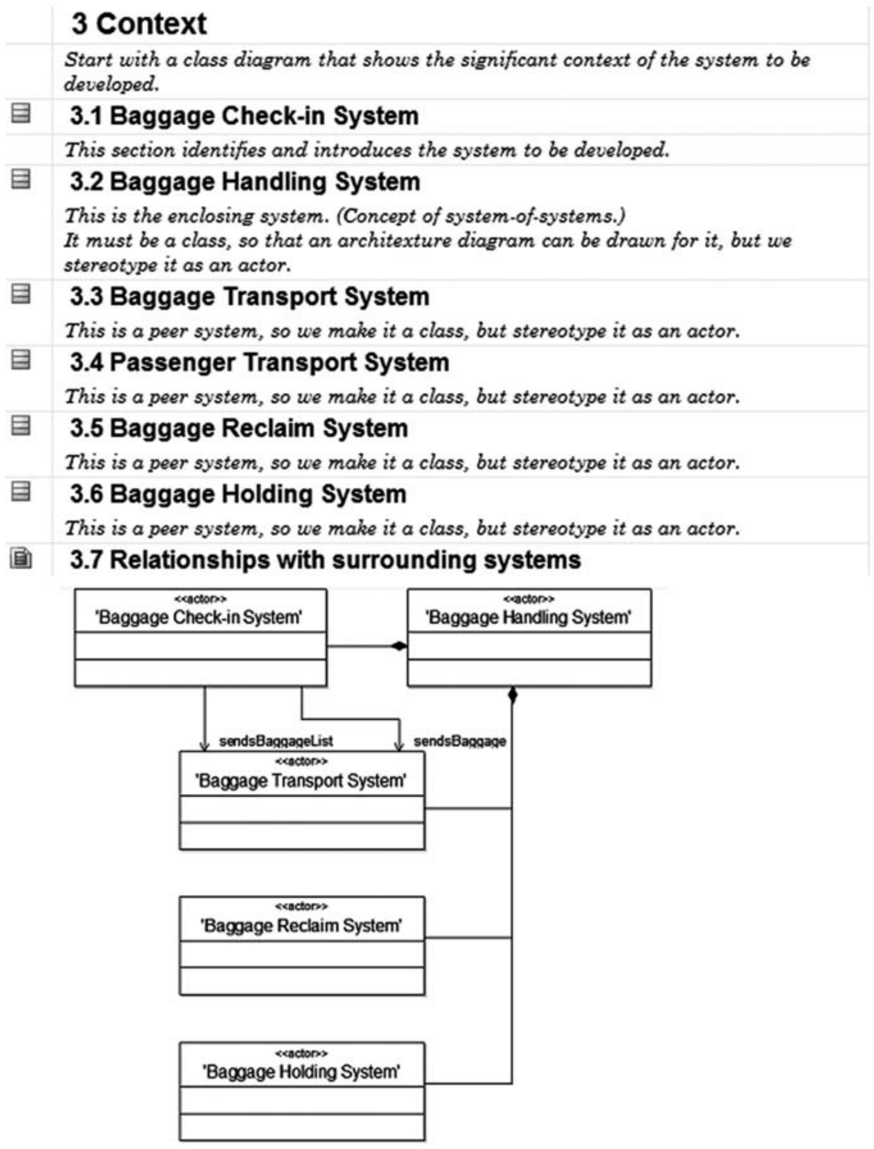
Optional class diagram showing subtype relationships between stakeholders, if any.

Figure 7.14 Stakeholders section of design document.

are modelled using aggregations and associations. Again, each class and association appears in the design document with a textual description.

- *Usage.* This section describes the top level use cases for the system. This is presented as a series of use case diagrams, each with one or more sequence diagrams. Figure 7.16 shows just one of the use cases and its sequence diagram showing the normal course of action for the scenario. The sequence diagram shows the interactions between the stakeholders (some of which are external subsystems) and the system in question (the baggage check-in system) and thus helps to define the scope, process context and external interfaces.
- *Design rationale.* This section summarizes the analysis and modeling activity by giving an explanation of how the need is going to be satisfied by the capabilities of the system. One way of presenting this information is in the form of a "satisfaction argument" for each statement in the input requirements document. It is here that the traceability to high-level requirements and from low-level requirements is established. The satisfaction argument, in effect, explains how the statement of need has been decomposed into statements of capability. This is illustrated in Figure 7.17.

In this figure, the first column shows the text of the statement of need that is addressed by the rationale, the middle column contains the rationale and the right-hand column shows evidence for the rationale in the model and requirements that are derived from it. This tabular presentation is, in effect, the sandwich on its side: two layers of requirement with the design rationale in between. With effective tool support, this view of the project data can be generated from the presence of tracing between the layers.



3 Context

Start with a class diagram that shows the significant context of the system to be developed.

3.1 Baggage Check-in System

This section identifies and introduces the system to be developed.

3.2 Baggage Handling System

This is the enclosing system. (Concept of system-of-systems.)
It must be a class, so that an architexture diagram can be drawn for it, but we stereotype it as an actor.

3.3 Baggage Transport System

This is a peer system, so we make it a class, but stereotype it as an actor.

3.4 Passenger Transport System

This is a peer system, so we make it a class, but stereotype it as an actor.

3.5 Baggage Reclaim System

This is a peer system, so we make it a class, but stereotype it as an actor.

3.6 Baggage Holding System

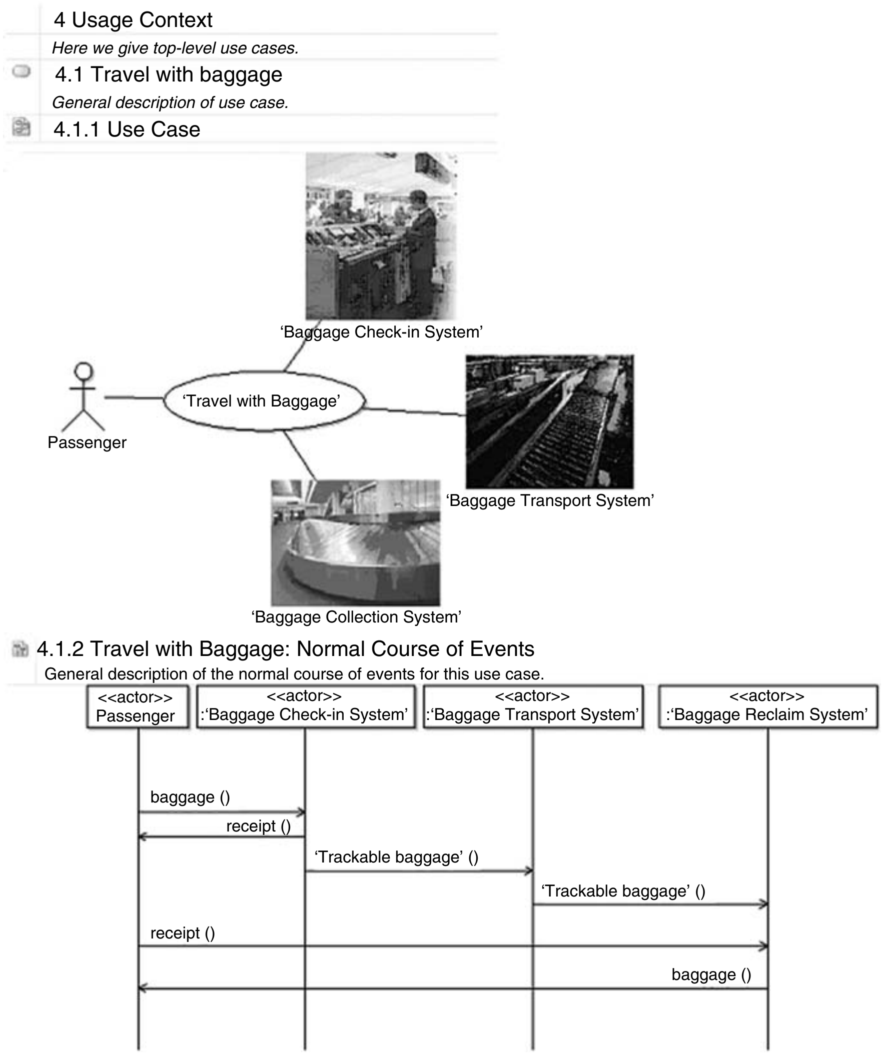
This is a peer system, so we make it a class, but stereotype it as an actor.

3.7 Relationships with surrounding systems

Figure 7.15 Context section of design document.

7.11 Metrics for Traceability

Since the concept of traceability is so central to requirements engineering, it is interesting to consider what process measurements may be useful in relation to the flow-down of requirements.



4 Usage Context

Here we give top-level use cases.

4.1 Travel with baggage

General description of use case.

4.1.1 Use Case

4.1.2 Travel with Baggage: Normal Course of Events

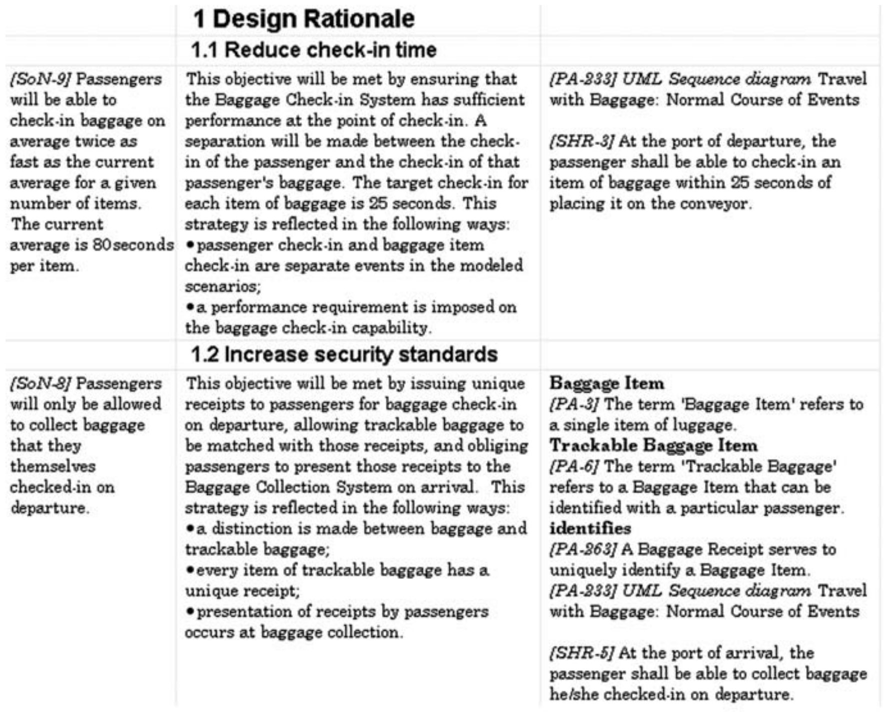
General description of the normal course of events for this use case.

'Baggage Check-in System'
Passenger
'Travel with Baggage'
'Baggage Transport System'
'Baggage Collection System'
<<actor>> Passenger
<<actor>> :'Baggage Check-in System'
<<actor>> :'Baggage Transport System'
<<actor>> :'Baggage Reclaim System'
baggage ()
receipt ()
'Trackable baggage' ()
'Trackable baggage' ()
receipt ()
baggage ()

Figure 7.16 Usage section of design document.

Focusing on the satisfaction relationship, and moving down through the layers of requirements, there are three dimensions of traceability that may interest us:

- ***Breadth*:** how well does the relationship cover the layer, upwards and downwards?
- ***Depth*:** how far down (or up) the layers does the relationship extend?
- ***Growth*:** how much does the relationship expand down through the layers?



	1 Design Rationale	
	1.1 Reduce check-in time	
[SoN-9] Passengers will be able to check-in baggage on average twice as fast as the current average for a given number of items. The current average is 80seconds per item.	This objective will be met by ensuring that the Baggage Check-in System has sufficient performance at the point of check-in. A separation will be made between the check-in of the passenger and the check-in of that passenger's baggage. The target check-in for each item of baggage is 25 seconds. This strategy is reflected in the following ways: • passenger check-in and baggage item check-in are separate events in the modeled scenarios; • a performance requirement is imposed on the baggage check-in capability.	*[PA-333] UML Sequence diagram* Travel with Baggage: Normal Course of Events *[SHR-3]* At the port of departure, the passenger shall be able to check-in an item of baggage within 25 seconds of placing it on the conveyor.
	1.2 Increase security standards	
[SoN-8] Passengers will only be allowed to collect baggage that they themselves checked-in on departure.	This objective will be met by issuing unique receipts to passengers for baggage check-in on departure, allowing trackable baggage to be matched with those receipts, and obliging passengers to present those receipts to the Baggage Collection System on arrival. This strategy is reflected in the following ways: • a distinction is made between baggage and trackable baggage; • every item of trackable baggage has a unique receipt; • presentation of receipts by passengers occurs at baggage collection.	**Baggage Item** *[PA-3]* The term 'Baggage Item' refers to a single item of luggage. **Trackable Baggage Item** *[PA-6]* The term 'Trackable Baggage' refers to a Baggage Item that can be identified with a particular passenger. **identifies** *[PA-363]* A Baggage Receipt serves to uniquely identify a Baggage Item. *[PA-333] UML Sequence diagram* Travel with Baggage: Normal Course of Events *[SHR-5]* At the port of arrival, the passenger shall be able to collect baggage he/she checked-in on departure.

Figure 7.17 Rationale section of design document.

To help in determining which aspects of these dimensions are useful in terms of measuring the requirements engineering process, it is necessary to distinguish between two types of metrics:

- ***Phase metrics*****:** measurements relating to a single stage of development, *e.g.* just to the systems requirements layer.
- ***Global metrics*****:** measurements spanning several stages of development.

The three dimensions are now addressed, along with a discussion about balance.

7.11.1 Breadth

Breadth relates to coverage and as such is a phase metric. As discussed in Chapter 1, coverage can be used to measure progress of processes that create traceability at a single stage. It focuses on a single layer and measures the extent to which requirements are covered by the adjacent level above or below (or "beside" when looking at qualification.)

7.11.2 Depth

Depth looks at the number of layers that traceability extends upwards or downwards from a given layer, making it a global metric. One application may relate

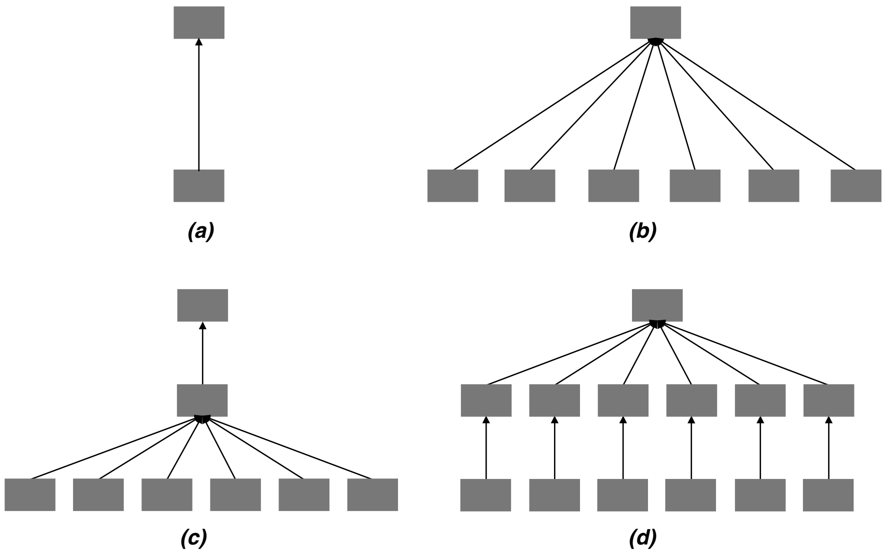


Figure 7.18 Traceability growth.

to determining the origins of requirements of the lowest level. How many component requirements have actually flowed down all the way from the stakeholder requirements, and how many have their origin somewhere in the design?

7.11.3 Growth

Growth is more interesting. It is related to potential change impact. How many requirements at lower levels are related to a single requirement at the top level? Consider Figure 7.18, in which four situations are contrasted.

In case (a), a single requirement is satisfied by a single requirement at the next level down. The growth factor is 1. In (b) the single requirement is met by 6, giving a growth factor of 6. What does this say about the differences between the two requirements? Possibilities are:

- requirement (b) may be poorly expressed, and needs decomposing into several;
- requirement (b) may be inherently more complex than (a), and therefore may need special attention;
- changing requirement (b) will have more impact than changing (a), and therefore needs special attention.

Of course, an apparent imbalance at one level may be addressed at the next level down. This is illustrated by cases (c) and (d), where the growth factor two levels down is identical. What could be deduced from this? Possibilities are:

- the top requirement in (c) was at a level too high;
- the middle requirements in (d) were at a level too low.

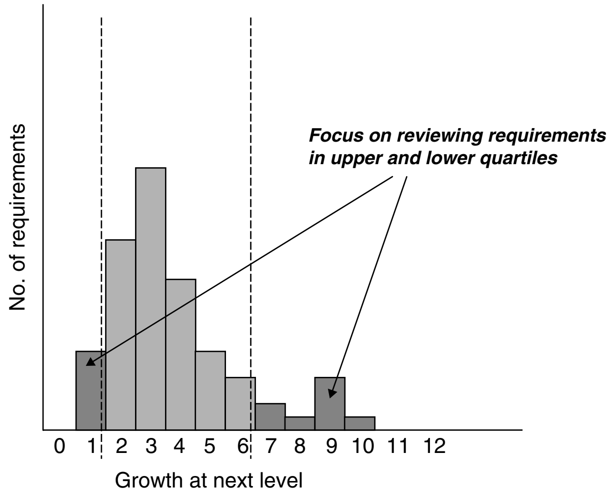


Figure 7.19 Frequency distribution of requirement growth.

Only after considerable experience in a particular organization developing particular kinds of systems could one begin to ascertain what growth factor of requirements between layers is to be expected. More readily useful, however, would be to examine the balance of growth between requirements as a means of identifying potential rogue requirements, or imbalances in the application of process.

7.11.4 Balance

One idea for a metric is to look at the distribution of growth factors for individual requirements between two given layers, and examine those that lie in the outer quartiles of the distribution. The goal is to identify requirements that have an abnormally high or low growth factor, and subject them to special scrutiny.

Figure 7.19 shows what a typical growth distribution may look like. The graph plots the growth rate against the number of requirements that possess that growth rate. Most lie between 2 and 6, whereas a few have only 1 or more than 6. It is these latter requirements that should be identified and given special attention.

The discussion above was about downwards growth – examining the number of requirements that flow out of another. What about the opposite direction: the number of requirements that flow *into* another?

Bearing in mind that traceability is a many-to-many relationship, consider Figure 7.20. Two requirements at the lower level have more than one requirement flowing into them. What can we say about these requirements? They are perhaps more critical than others, since they satisfy multiple requirements, and should therefore be given special attention.

The distribution of upward traceability can be used to single out these requirements. Figure 7.21 shows the typical shape of such a distribution.

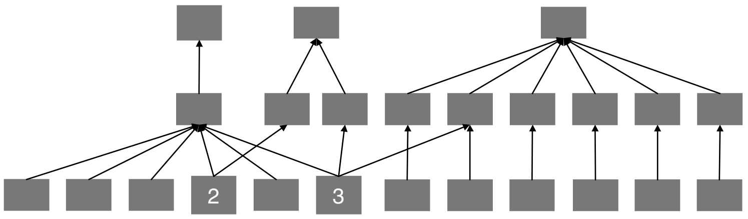


Figure 7.20 Criticality of requirements.

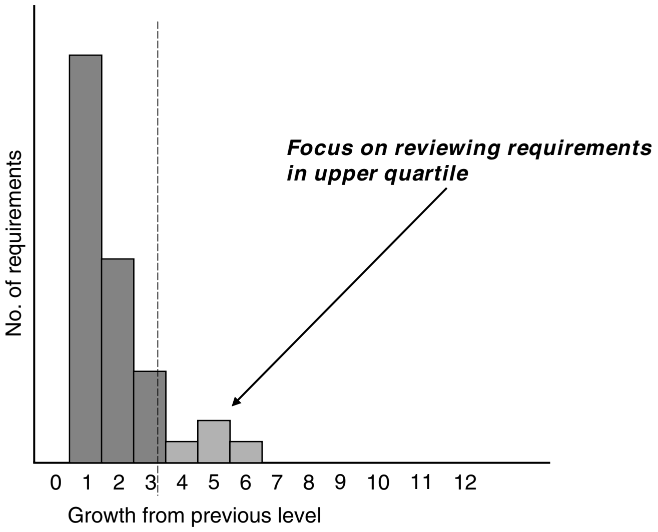


Figure 7.21 Frequency distribution of requirement criticality.

7.11.5 Latent Change

Change management is perhaps the most complex requirements engineering process. The processes and information model detailed in Chapter 2 take advantage of traceability to determine the potential impact of change. When a change request is raised against one requirement, all those tracing to it move to a suspect status until the engineers ascertain the true impact.

The raising of a single change request, therefore, can suddenly introduce a cascade of potential latent change into the system. In such circumstances, it would be highly desirable to track progress and estimate the consequential work.

Figure 7.22 illustrates the complexity of change impact. A change request is raised on one of the highest level requirements. Part (a) shows the potential impact using downwards traceability. Those boxes marked with a white circle are subject to change assessment.

Part (b) shows the potential change using upwards impact. This occurs because of a low-level requirement that flows down from two higher requirements. It is necessary to access upwards impact from these changes, because

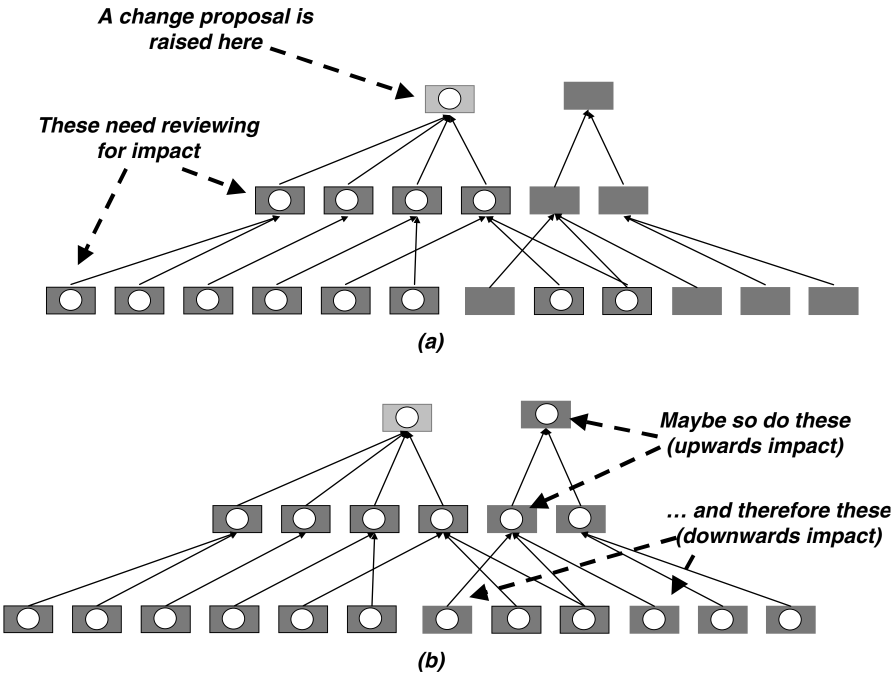

A change proposal is raised here
These need reviewing for impact
(a)
Maybe so do these (upwards impact)
... and therefore these (downwards impact)
(b)

Figure 7.22 Potential change resulting from a change request.

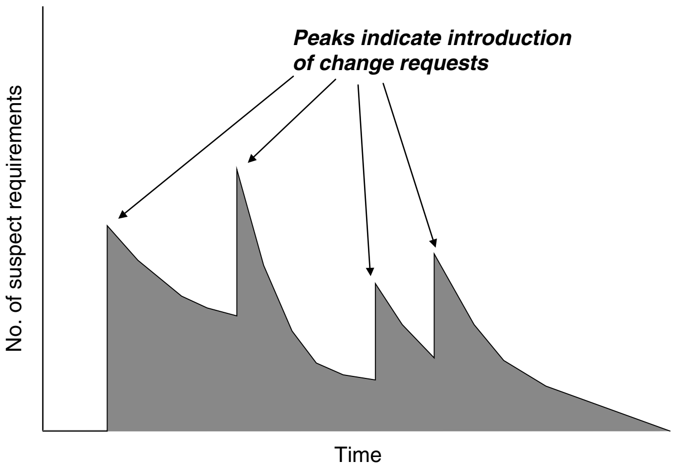


Figure 7.23 Progress in processing change.

changes in a low-level requirement may cause renegotiation at a higher level. Suddenly *everything* in this example is potentially subject to change!

Of course, as engineers assess the real impact, it may be found that in fact some of these requirements are not subject to change after all, and the cascade of potential changes can thankfully be pruned, sometimes substantially.

The status of change can simply be measured in terms of the number of requirements still in a suspect state. When a change request is raised, all other requirements traceable downwards and upwards are marked as suspect. Then the number of suspect requirements will steadily decrease as assessments are made of each, their state is reset, possibly resulting in a cascade of others also being reset. The amount of residual change in a system will therefore peak every time a new change is introduced, and tail-off, as illustrated in Figure 7.23.

The above discussion of the change process supposes that change is propagated from requirement to requirement purely through the *existing* set of links. However, a change in a requirement may necessitate the addition or removal of traceability links. Changes in links should propagate change to the connected requirements at both ends.

7.12 Summary

Of all the advantages in the use of traceability cited in Section 1.5, it is the increase in confidence in meeting requirements that is so clearly addressed through rich traceability. The discipline of capturing the rationale associated with traceability builds that confidence.

There is no doubt that there is considerable effort involved in the creation of complete satisfaction arguments, especially in complex systems with hundreds of requirements.

In the Network Rail project, there are some 500 satisfaction arguments that serve to decompose the high-level requirements through to subsystem requirements. A team of between two and five requirements engineers was dedicated to the maintenance of this information over about 3 years.

Experience suggests, however, that the cost is amply repaid in the increased confidence that comes from the greater reflection required. The ability of the Network Rail sponsor organization to take a high-level objective and demonstrate in detail through the layers of rich traceability exactly how that objective is going to be met was a major selling point for the concept.

It is clear, also, that traceability is a rich source of metrics for process measurement. It is the formalization of relationships through traceability and associated processes that makes such measurement possible.