www.telelogic.com

Rich Traceability

by Jeremy Dick

Quality Systems and Software Ltd.

Abstract

Creating traceability through the use of links between paragraphs of documents, or between objects in a requirements database, is a great step forward for many system engineering organisations. It enables analysis of impact, coverage and derivation to be performed, and greatly improves the requirements management process.

However, the semantics of the relationship induced by such links is very shallow: when one object is linked toanother, the strongest meaning it can have is “this object may be impacted by changes to the other object”.

The next step in process improvement through traceability may be to encourage the use of a deeper semantics in the traceability relationship. Satisfiability, for instance, is a richer relationship, requiring the ability to explain that one user requirement may be satisfied by the conjunction of several system requirements, or by any one of a set of system requirements.

This paper builds on the ideas o f others to define a possible approach to richer traceability relationships, making use of textual rationale and propositional logic in the construction of traceability arguments. The underlying logic allows other, deeper kinds of analysis to be performed.

The also paper discusses the way the same structures can be applied to the management of requirements for product families. The use of “exclusive or” in rich traceability provides a way of representing alternative ways of meeting sets of requirements. It can therefore be used to represent the variance in system requirements addressed by different configurations of a product range.

1 Introduction

The creation and analysis of traceability has become a key practice in system engineering. By linking requirements in one document with requirements in another, the relationship between the two can be documented. This relationship can then be used for various kinds of analysis, including:

- ***Impact analysis***. What artefacts might be affected if a given requirement changes?
- ***Derivation analysis***. What requirements have given rise to the need for a given artefact?
- ***Impact coverage analysis***. Have all requirements been taken into account?

- ***Derivation coverage analysis***. Is there a need for every artefact?

Working from the requirements downwards, impact coverage analysis can be used as a measure of progress in the construction of a response to requirements, the goal being to achieve 100% coverage. Working in the other direction, derivation coverage analysis can be used to guard against "gold plating": are all aspects of the design covered by requirements?

The ability to carry out these kinds of analysis represents a quantum leap improvement in the requirements engineering process for many organisations. For instance, by using impact coverage analysis, one system test team was able to discover that their system test plan covered only 78% of the system requirements. This knowledge enabled them to improve the system test process, guided by the vital impact coverage measure.

The idea of a compliance matrix, often requested as evidence for the quality of a response, is a summary of a traceability relationship.

However, the semantics of such links is typically very shallow. For instance, if a number of system requirements are linked to a user requirement, what do the links signify? Does it mean that *all* the linked system requirements are necessary to satisfy the user requirement? Or does it mean that any one of them is necessary?

Most usually, the strongest claim that can be made for traceability links is that they indicate *possible impact*: i.e. these system requirements are likely to be impacted by a change in the user requirement.

This paper is about techniques for creating deeper kinds of traceability, and the richer forms of analysis that are thus permitted.

2 Requirements for rich traceability

Several things are desirable for creating more meaningful traceability relationships:

- ***Textual rationale.*** In addition to the links themselves, the ability to attach rationale, even just a textual description of the reason for links, would assist in determining the exact relationship being described.
- ***Grouping of links.*** Often single links are insufficient to portray the deeper meaning of a relationship. The ability to group links together for the purpose of explaining the relationship would also assist in describing the relationship.
- ***Typing of link groups.*** A more formal typing and structuring of groups of links would assist in providing new forms of analysis of traceability relationships. For instance, grouping of links might mean that they are mutually conflicting, or identical, or together form a response, or represent alternative responses.

3 An approach to rich traceability

Suppose three system requirements participate in responding to a single user requirement. Figure 1 portrays the expression of this relationship using simple traceability links.

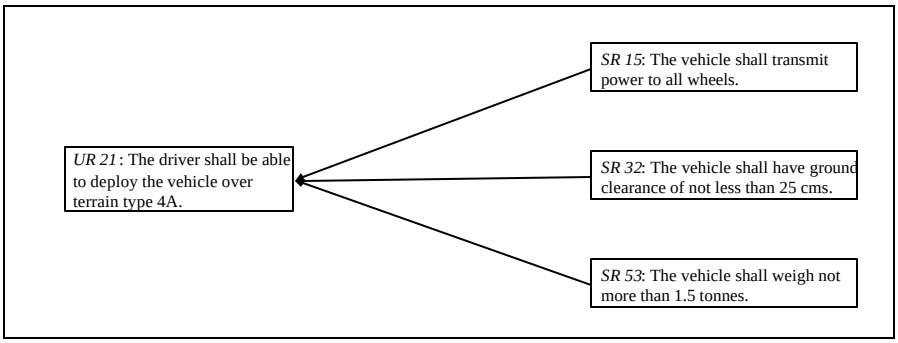

Figure 1 : Simple traceability

This relationship can be enriched by using a structure portrayed in Figure 2.

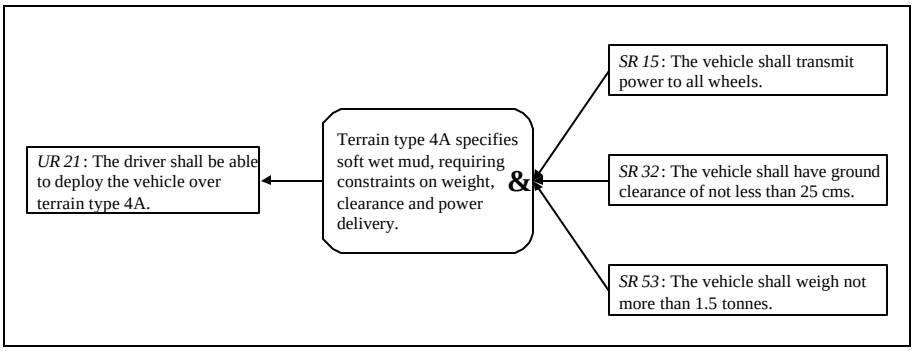

Figure 2: Rich traceability - conjunction

Here the relationship carries a textual rationale, and the three links are grouped as a conjunction, indicating that all three system requirements are required to satisfy the user requirement.

Richer structures can be deployed to represent alternative solutions. It may be that any one of several requirements on the system is sufficient to meet a user requirement, but that several of those system requirements may be present. Figure 3 depicts such a situation.

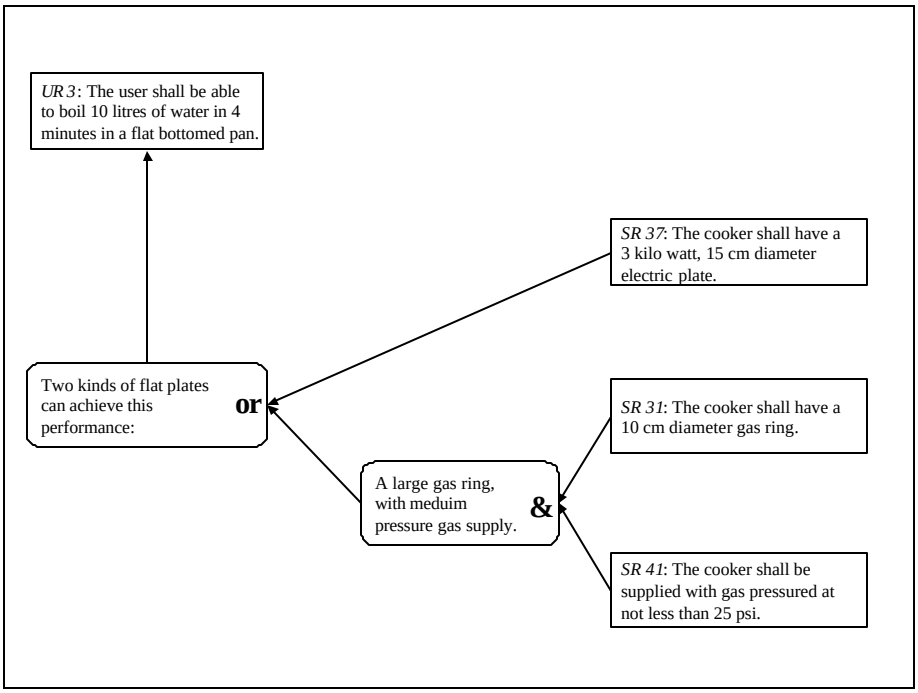

Figure 3 : Rich traceability – non -exclusive disjunction

In this example, the cooker may perfectly well have both an electric plate and a gas ring. Other requirements may exist that mean that both are necessary, or that only one is possible.

Another possibility is that one or more mutually exclusive solutions exist, as portrayed in Figure 4.

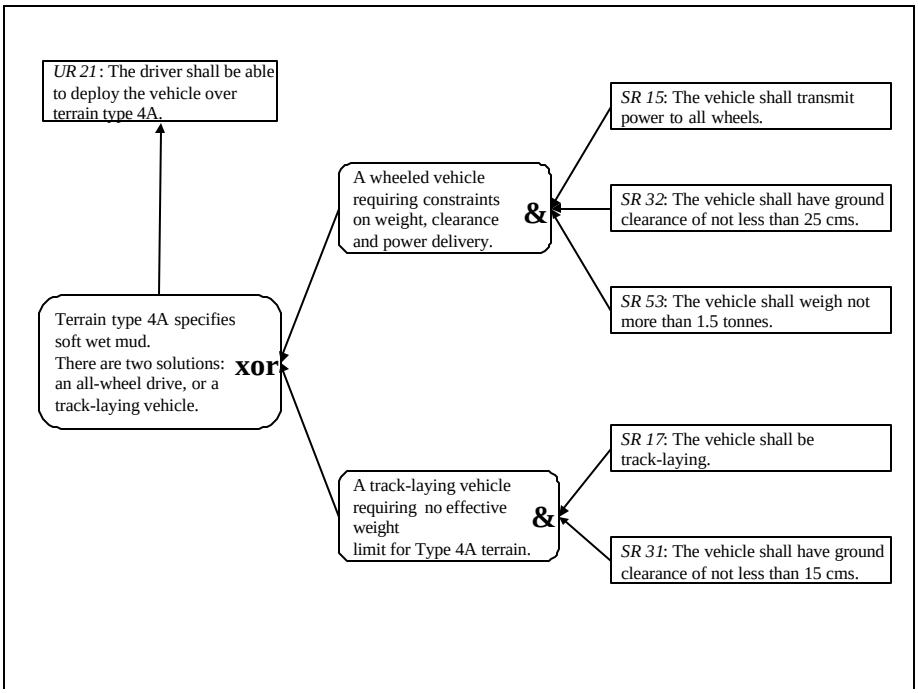

Figure 4 : Rich traceability – exclusive disjunction

Here two solutions are depicted as two separate conjunctions of requirements, which are then grouped in an exclusive disjunction, indicating that only one of the solutions can be selected. It may be useful to delay the choice of solution and represent both possibilities, since other constraints may rule out one of the solutions at a late stage of the construction of the response.

4 Assumptions and Domain Knowledge

It is not only requirements that play a role in satisfaction arguments; we may wish also to capture assumptions and other information regarding the working domain. Figure 5 shows the inclusion of an assumption and domain knowledge.

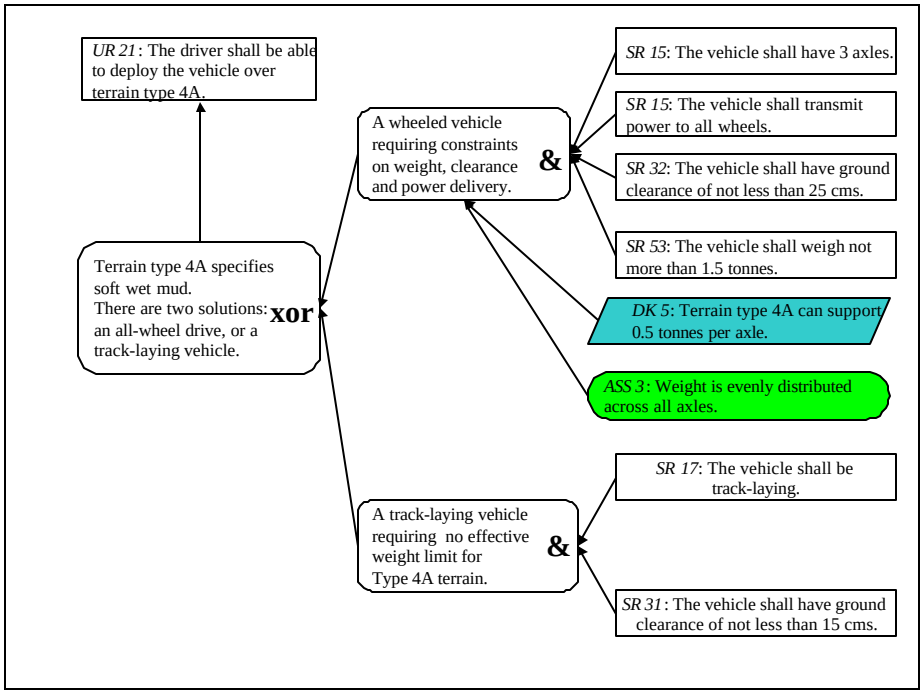

Figure 5 : Recording assumptions and Domain Knowledge

5 Multi-layered Rich Traceability

The examples used above have focussed on the task of creating a response to a set of user requirements. The same principles can be applied to creating a design response to system requirements, or a validation and verification response to system requirements, or at any level of the systems development process.

Figure 6 shows rich traceability through two layers: user requirements to system requirements, and system requirements to design requirements.

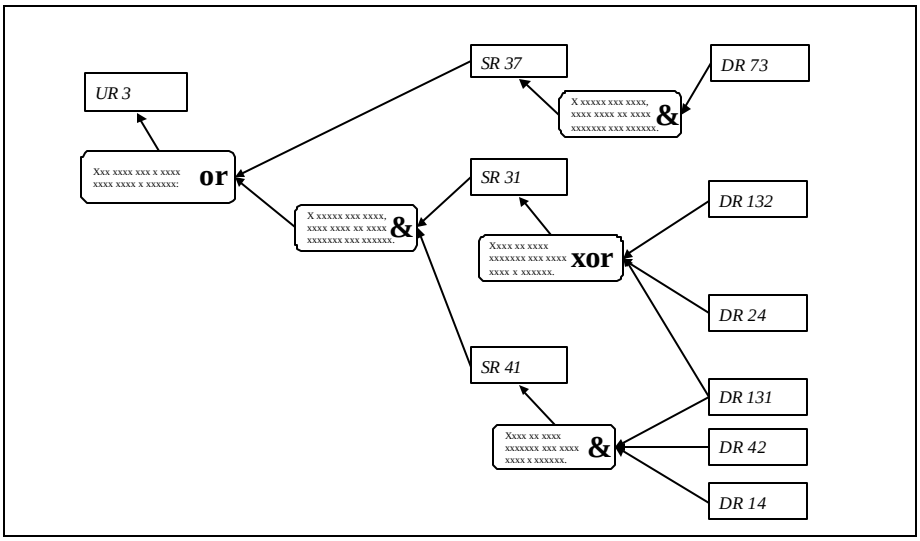

Figure 6: Rich traceability through two layers

6 Analysis of rich traceability

6.1 Traceability analysis

The four kinds of traceability analysis mentioned in Section 1, *Introduction,* remain valid for this richer kind of relationship, only in every case the analysis is better informed.

For instance, not only can we distinguish those system requirements impacted by possible changes in a user requirement, but the propositional structures in place give us detailed information about the nature of the impact.

Similar arguments apply to derivation and both kinds of coverage analysis.

6.2 New kinds of analysis

The picture that is emerging through the above examples is the use of structures that use textually supported propositional combinators to document the exact nature of the relationship between a set of requirements and the response to them.

The full propositional calculus does not seem to be at play here, since there is no obvious role for universal or existential quantifiers in the structures proposed. This means that the forms of analysis required are a subset of classical propositional reasoning.

The advantage of using propositional structures in the way described above is that classical propositional reasoning can be used in the analysis of the resulting traceability relationship.

For instance, where exclusive disjunction (also known as “exclusive or”) is used to represent a graph of alternative solutions, simple proposition logic can be used to determine the overall consistency of design choices made in constructing a response.

Following the example of Figure 4, a choice of one branch induces the proposition

(SR15 & SR32 & S53) & ¬(SR17 & SR31)

and the other branch the proposition

¬(SR15 & SR32 & S53) & (SR17 & SR31)

where ¬*S* is read as "*not S*". No choice of alternative solutions should be made which introduces both *S* and ¬*S* into the set of system requirements; otherwise an inconsistency will result.

This kind of analysis could be called "consistency analysis". It is similar to the kind of analysis carried out by Binary Decision Diagrams [BDDs], used in hardware and software verification.

Where rich traceability is used through several layers of development, the question arises as to how the transitivity of the relationships could be exploited. In fact, the propositional structure extends quite uniformly through the layers, the intermediate artefacts forming conjunction nodes.

Return for a moment to Figure 6. If we consider the leaf *DR* requirements as Boolean constants and the *SR* requirements as conjunction nodes, the propositional expression induced by the whole structure amounts to

(A) *DR73 or (((DR132 & ¬DR24 & ¬DR131) or*
(¬DR132 & DR24 & ¬DR131) or
(¬DR132 & ¬DR24 & DR131)
) & DR131 & DR42 & DR14
).

Using the rules of propositional logic, this expression reduces to

(B) *DR73 or (¬DR132 & ¬DR24 & DR131 & DR42 & DR14),*

since *DR131* appears in a conjunction attached to *SR41* as well as the exclusive disjunction attached to *SR31*, thus forcing the particular choice.

The same result can be achieved by analysing the structure layer-by-layer. The propositional expression derived from the structure down to the *SR* requirements (treating them as the leaf Boolean constants) is

(C) *SR37 or (SR31 & SR41).*

Now we can repeat the analysis on the next layer down by replacing each *SR* constant in this expression by the propositional sub-expression derived from the sub-structure below it. In doing so, we arrive at expression (A) above, demonstrating the transitive nature of the analysis.

7 Applications

Rich traceability can be applied wherever simple traceability is applied today. Particular mention should be made about the following tasks.

- *Responding to an Invitation to Tender (ITT) or Request For Proposal (RFP).*
  The process of creating a tender against an ITT can be greatly enhanced by using rich traceability. It is used to construct a detailed argument for compliance, and informs the potential supplier as well as the customer as to the nature of the response in a formalised way that out-performs the classical request for a "compliance matrix".
- *Assessing a response to an ITT or RFP.*
  The assessment of a response requires reasoning about how the proposal meets a pre-determined set of criteria. Rich traceability can be used to construct the argument. Of course, the criteria may already be related to the requirements (rich traceability established by the customer), and the requirements similarly related to the proposal (rich traceability established by

the proposer). Then the transitive relationship may give sufficient information for an appropriate assessment.

8 Application to product family management

A key problem in managing a product family is to know which of a range of products or product features is best suited to a given set of requirements. The structures proposed in rich traceability offer the ability to represent a product family as a set of possible responses to a set of possible requirements, giving the possibility of managing the product family at the requirements level rather than the component level.

Ground work has been done in this area in [MKKW], where structures are proposed which encode the product variants at the requirements level. In one application, 328 key requirements were placed into a structure which reduced the number of choices to be made to 9.

Layers of rich traceability push this idea further by allowing the relationships between design and requirements to be included into the process, so that selection of requirements will induce selections of designs and solutions.

The top level description of the product family is the set of all possible system requirements met by the family of products. Some of these requirements may be universal to every application of the product family. Others will only be present in some applications; indeed, some requirements in the set will be mutually exclusive.

So our starting point is to build a rich traceability structure *on top of* the top-level requirements, which will enable us to represent the choices of requirement that are available. This is illustrated in Figure 7 with a partial structure for part of a family of cooker products.

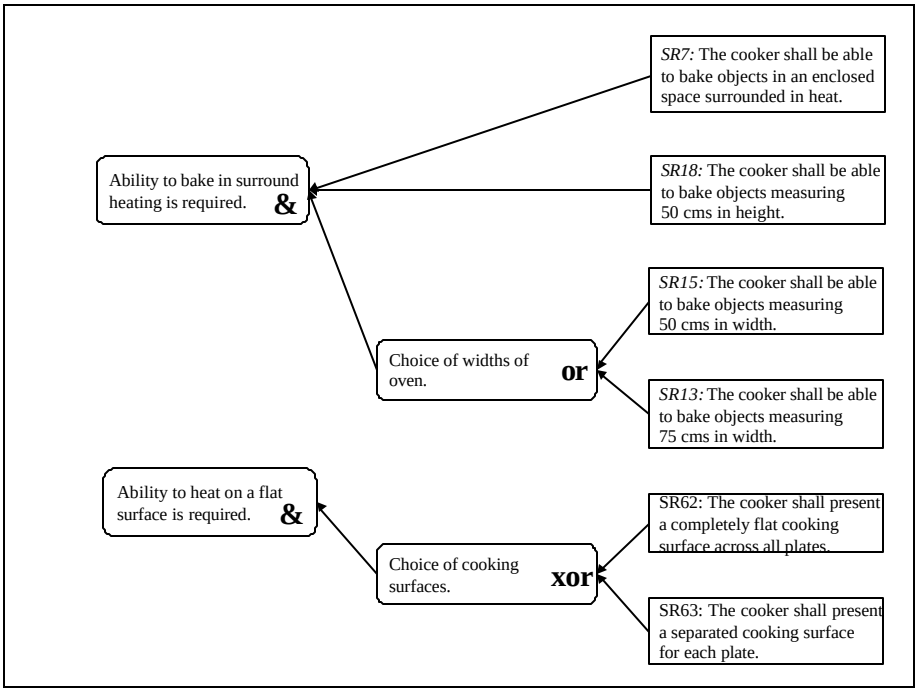

Figure 7: Top-level requirements choices

Notice that this layer of rich traceability is incomplete, in that it traces between the system requirements and nothing. The process of instantiation of the product

family consists in taking the user requirements, and completing the top layer of traceability by linking the user requirements to the system requirements through the traceability structure. As this occurs, a coherent and consistent choice of system requirements is controlled by the propositional structure. A example of this is presented in Figure 8.

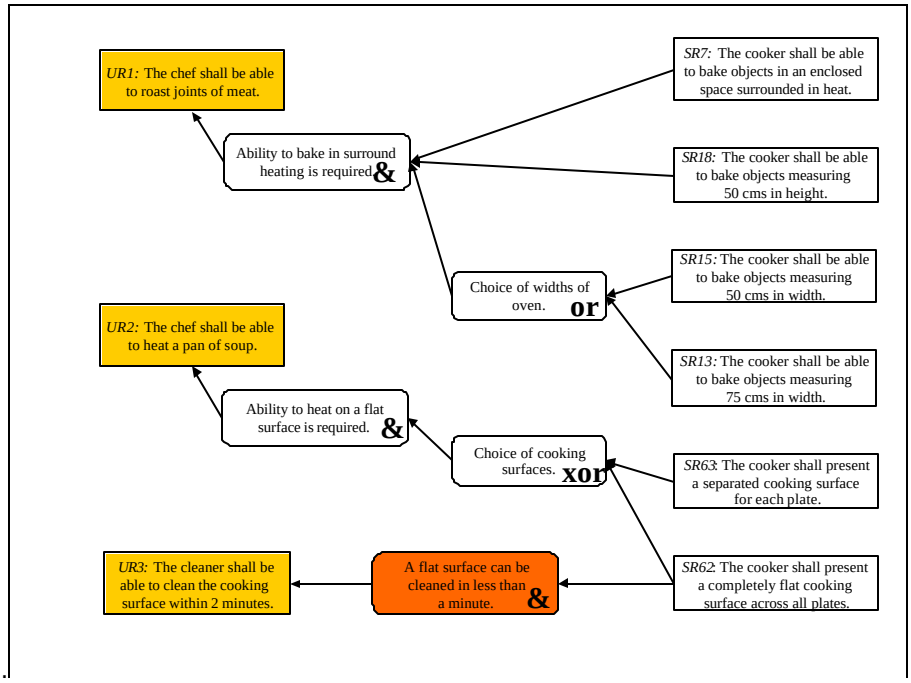

Figure 8 : Instantiating a product family against user requirements.

The underlying propositional structure can be analysed to discover the remaining degrees of freedom that exist in the product family for meeting the requirements. In this case, the resulting propositional expression is

(C) *(SR7 & SR18 & SR15 & SR62) or (SR7 & SR18 & SR13 & SR62)*

reflecting the fact that the only remaining decision revolves around the choice of oven width. The choice of cooking surface has been forced by the constraint on cleaning time.

If the rich traceability is continued into the design levels, then choices will also be enforced at that level, resulting, perhaps, in the selection of a single, or small set of possible members of the product family that will meet the user requirements.

More realistically, it may be found that no existing member of the product family currently meets the user need, and a variant will have to be built. This can only be because a conflict has been generated in the propositional structure through the choice of top-level requirements. It is possible that the rich traceability structure can then be used to identify which requirements are not being met. This information, in turn, could be used to derive requirements for a variant product.

9 Conclusions

Rich traceability is a simple idea that enables more detailed arguments for compliance to be constructed in a formalised way. It also provides a technique for the representation of alternative solutions to statements of requirements, which seems to have an important application in the management of reuse in product families at the requirements level.

In particular, rich traceability can provide the basis for:-

- the instantiation of a product family against a set of user requirements;
- cascading the selection of product family abilities down to the component level;
- the identification of places where the product family can best be enhanced, or a variant created, to meet a new need;
- the semi-automatic provision of an argument as to how a product responds to set of user requirements.

Acknowledgements

The ideas expressed in this paper are based on concepts used in the REVEAL method, a trademark of Praxis Critical Systems. These have been further developed in discussions with other QSS consultants and systems engineers from QSS customers, particularly Anthony Hall of Praxis Critical Systems and Jean-Claude Fromenteau of Thompson CSF.

References

[MHK] M Mannion, D Harper, B Keepence, 'Domain Requirements Definition using Viewpoints', IEEE Software, Jan 1998, Vol 15, No. 1, pp95-102

[MKKW] M Mannion, B Keepence, H Kaindl (Siemens), J Wheadon, 'Reusing Single Requirements From Application Family Requirements', 21st IEEE International Conference on Software Engineering (ICSE'99), May 1999.

[BDDs] … Madre ….