Rich Traceability

Jeremy Dick
TelelogicUK, Northbrook House, Oxford Science Park, OX10 7JL, UK
jeremy.dick@telelogic.com

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 systems 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 to another, 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 of 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 paper also 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 analyses that are thus permitted.

2. Origins and related work

Rich traceability draws together ideas from a number of sources. Firstly, the concept of a "design justification" as applied in the REVEAL method, a trademark of Praxis Critical Systems. In this approach, information is collected to justify traceability, and the

concepts of conjunction and disjunction are used to characterise the way in which design requirements combine to satisfy the high-level requirements. A design justification may itself have a hierarchical structure, with arguments and sub-arguments.

Secondly, the concept of "elaboration" is describes in the MoD SMART procurement process. Here a single statement explaining how the requirement is satisfied accompanies each requirement, and represents a simple form of rich traceability.

Thirdly, the Goal Structuring Notation [GSN], which is used for the construction of safety arguments, makes use of structures very similar to those of rich traceability. Safety goals are systematically decomposed using simple propositional logic, contextual assumptions identified, and supporting evidence collected.

Another tool-supported approach to goal-directed requirements analysis is described in [Kaos].

3. 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, assists 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 also assists in describing the relationship.
- ***Typing of link groups.*** A more formal typing and structuring of groups of links assists 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.

4. 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 elementary traceability links.

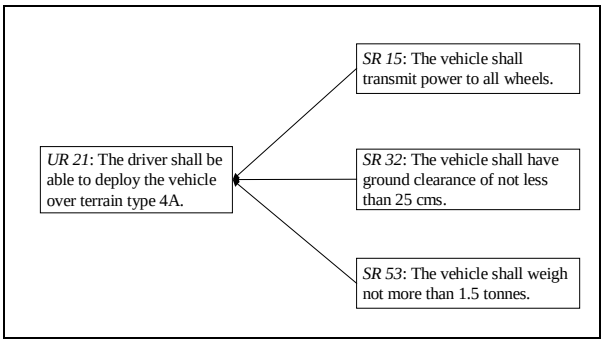

Figure 1: Elementary traceability

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

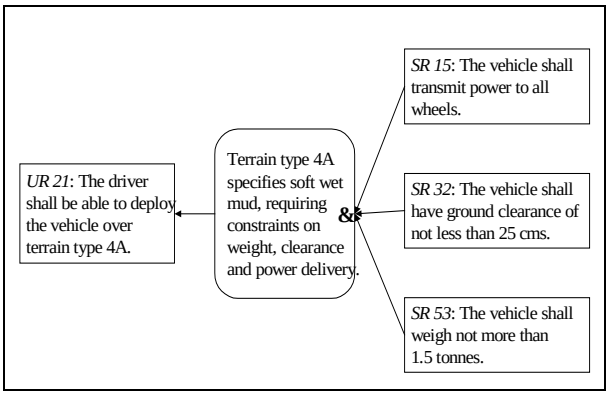

Figure 2: Rich traceability - conjunction

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.

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.

It is not only requirements that play a role in satisfaction arguments; we may wish also to capture other information regarding the working domain.

Capturing domain knowledge so explicitly is a good discipline. It clarifies, and lays open to review, the assumptions that are being made in the satisfaction argument.

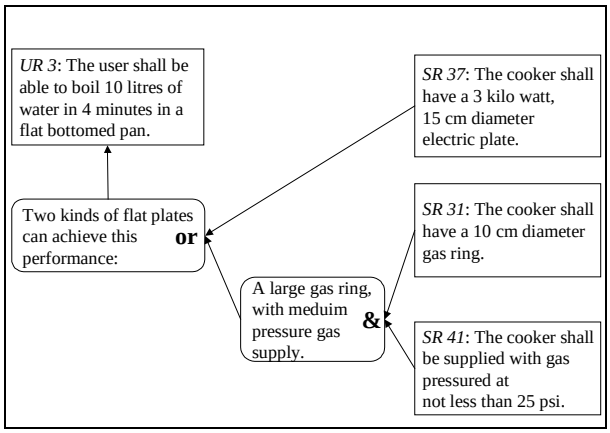

Figure 3: Rich traceability – non-exclusive disjunction

5. Analysis of rich traceability

5.1 Traceability analysis

The four kinds of traceability analysis mentioned in Section 1 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 also 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.

5.2 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.

One kind of analysis would be to use the propositional calculus to reduce the graphs to expressions in a normal form that indicates the number of degrees of freedom, or possible solutions, that remain in a rich traceability structure.

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

6. 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.

7. 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.

Groundwork 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 4 with a partial structure for part of a family of cooker products.

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. An example of this is presented in Figure 5.

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:

(SR7 & SR18 & SR15 & ¬SR13 & SR62) or (SR7 & SR18 & ¬SR15 & SR13 & SR62) or (SR7 & SR18 & SR15 & SR13 & SR62)

This expression reflects the fact that the only remaining decision revolves around the choice of oven width (or the provision of 2 ovens, one of each 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.

8. Implementation

This section describes ways in which rich traceability can be implemented. Although the requirements management tool DOORS is used as a vehicle for the discussion, the principles can be applied to other tools as well. No knowledge of DOORS is assumed.

8.1 Terminology

The following terminology for rich traceability is proposed.

- ***Established requirement***: the requirement being satisfied by the argument (UR3 in Figure 6.)
- ***Contributing requirement***: a requirement contributing to the argument (SR37, SR31, SR41 in Figure 6.)
- ***Main argument***: the principal rationale of the argument (the box marked "or" in Figure 6).
- ***Sub-argument***: a subsidiary rationale for an argument (the box marked "&" in Figure 6).
- ***Combination***: whether the argument is a conjunction, disjunction, etc.

This terminology is illustrated by Figure 6.

Before considering the best implementation of rich traceability, we should consider what we want to use it for. Broadly speaking, we want to be able to:

1. create hierarchical arguments between two identifiable layers of requirements
2. edit the rationale of the argument
3. designate the propositional type of the argument ("&" or "or")
4. navigate between requirements related by arguments
5. publish a satisfaction argument report.

8.2 Implementation approach

A limited form of rich traceability can be implemented quite simply by placing a text attribute on every established requirement, and recording in it a rationale for the requirement's tracebility links. However, this approach denies some of the "richness" of rich traceability. In particular, requirements 1 and 3 above are not satisfied.

Here is a more complete approach, used in the Railtrack West Coast Route Modernisation (WCRM) project, which seems to meet these requirements. Figure 7 shows the DOORS data model used, which described as follows:

Relationships

Two relationships are used:

- "establishes" for showing the links between main arguments and established requirements. This is usually a one-to-one relationship.
- "contributes to" for showing the links between contributing requirements (possibility in several different requirements documents) and arguments. This is a many-to-many relationship.

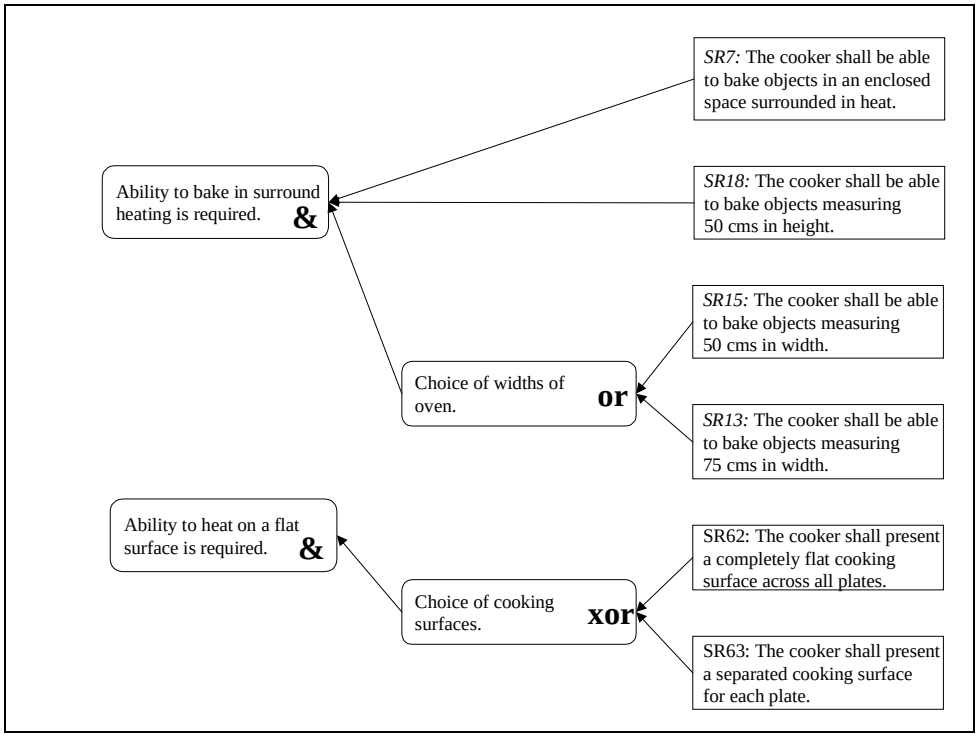

Figure 4: Top-level requirements choices

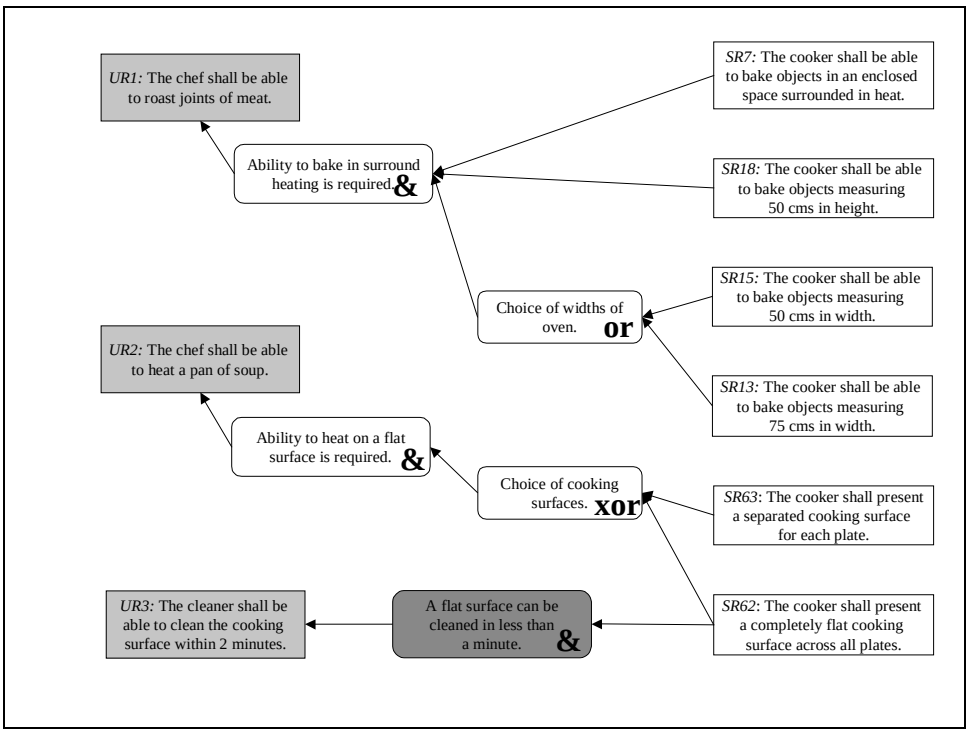

Figure 5: Instantiating a product family against user requirements.

Documents

For each layer of rich traceability, a "satisfaction argument" document is used to contain the nodes of the arguments as objects. The rationale is held in the "Object Text" attribute, and the type of structure as an enumerated attribute called "combination", taking values "&" or "or". The object hierarchy in the "satisfaction argument" document is used to represent the hierarchical structure of the argument.

Note that, whereas there is only likely to be a single document for established requirements in each layer, there may be multiple contributing documents.

Objects other than requirements may contribute to arguments, such as domain knowledge (assumptions about the world), or evidence obtained for system modelling or simulation.

Views

The key view for managing rich traceability is constructed in the Satisfaction Argument document. It has four columns, as show in Figure 8.

The first column uses a macro to follow the outgoing "establishes" link and display the identifier and text of the established requirement. The second column displays the "Combination" attribute, and is directly editable. The third column is the DOORS main column, and shows the rationale in the Object Text attribute, as well as any headings serving to structure the document. The rationale is thus directly editable.

The forth column uses a macro to follow each incoming "contributes" link, and display the identifier and text of the contributing requirements. This view can serve as a rich traceability report, as well as the focus for creating, reviewing and editing the satisfaction arguments.

Navigation to the related requirements is also possible by using the DOORS link indicators, outgoing links for established requirements, and incoming links for contributing requirements.

8.2.1 Other tools

A customization of DOORS specifically to handle rich traceability is under development by our professional services team. A key tool is the Rich Traceability Explorer which allows selective tracing through layers of Rich Traceability. A screen-shot of this tool is shown in Figure 9. The columns represent the layers of requirements, and the satisfaction arguments can be expanded to reveal their structure through the layers.

It is this tool that releases the real power of rich traceability by giving a view of requirements flow down across multiple layers. Requirements are marked by the "R" icon, and nodes in the satisfaction argument by "&" or "o" icons. The explorer allows a reviewer selectively to drill down through layers relationships and satisfaction arguments to understand how objectives are being met.

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.

References

GSN Andrew P Moore & J Klinker & David M Mihelcic, How to Construct Formal Arguments that Persuade Certifiers, In Michael G Hinchey & Jonathan P Bowen (Eds), *Industrial-Strength Formal Methods in Practice*, FACIT Series of Springer-Verlag 1999, pp285-314

Kaos Bertand, P. et al., Grail/Kaos: un outil pour l'analyse des besoins guidée par les objectifs, Génie Logiciels, 46, 12/97, pp 139-144

[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.

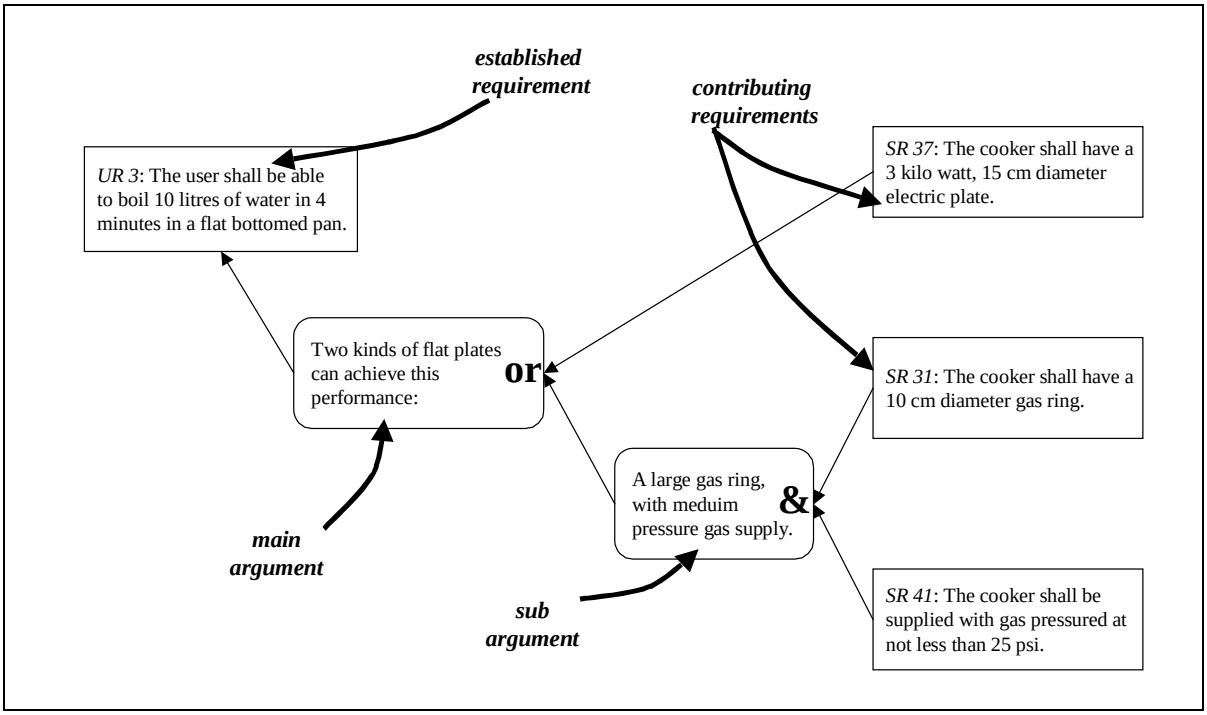

Figure 6: Terminology of Rich Traceability

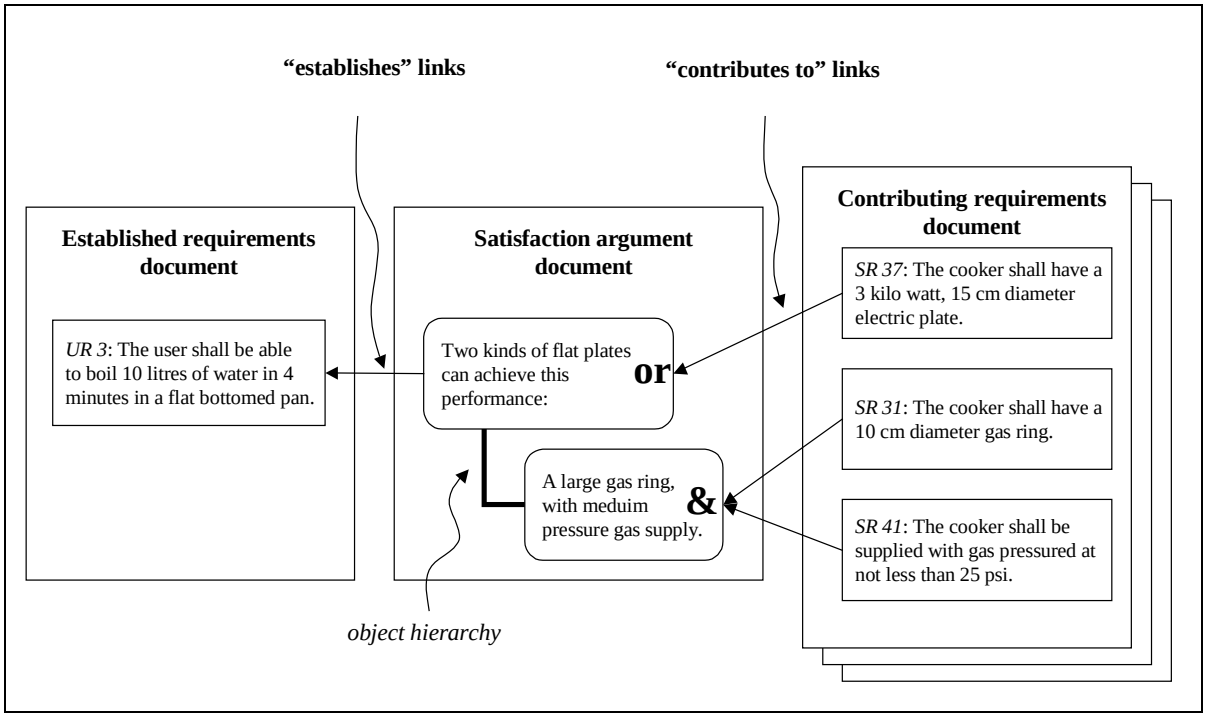

Figure 7: DOORS data model for Rich Traceability

1	2	3	4
Established requirement	**&/or**	**Satisfaction Argument**	**Contributing requirements**
UR 3: The user shall be able to boil 10 litres of water in 4 minutes in a flat bottomed pan.	**or**	Two kinds of flat plates can achieve this performance:	*SR 37*: The cooker shall have a 3 kilo watt, 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.
(next established requirement)		*(next satisfaction argument)*	

Figure 8: Tabular view of a satisfaction argument document.

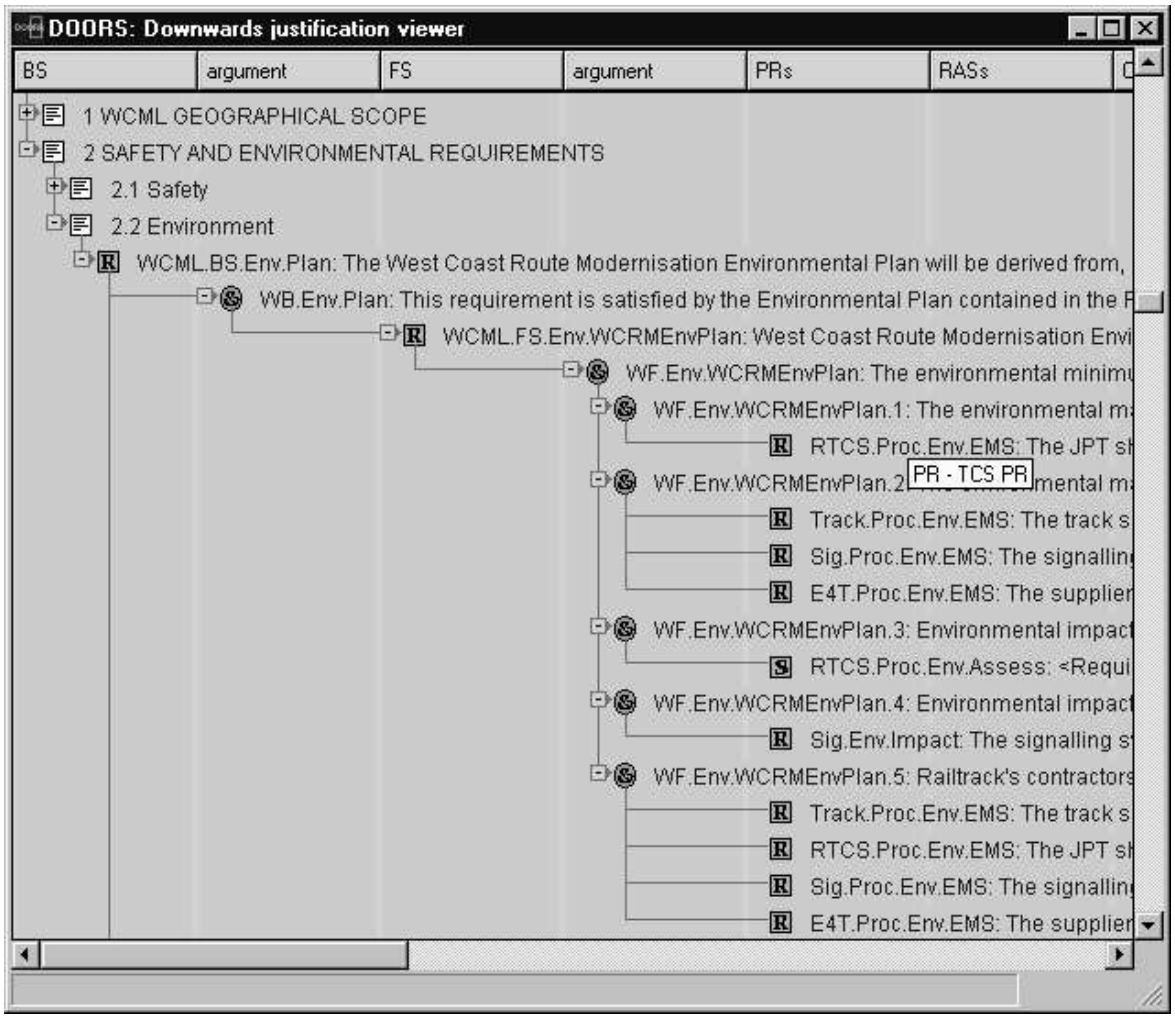

Figure 9: Screen-shot of the Rich Traceability Explorer