The Use of Satisfaction Arguments for Traceability in Requirements Reuse for System Families: Position Paper

Katrina Attwood, Tim Kelly, John McDermid

Rolls-Royce University Technology Centre in Systems and Software Engineering, Department of Computer Science, University of York, Heslington, YORK. YO10 5DD United Kingdom
{katrina.attwood, tim.kelly, john.mcdermid}@cs.york.ac.uk

1 Introduction

Refinement of requirements into specifications depends on the concept of requirements 'satisfaction'. This is a recursive process, in which the fulfillment of lower-level requirements is seen as a sufficient condition for the fulfillment of the higher-level statement [1]. Traceability structures record satisfaction relationships between requirements at various levels of abstraction. It is essential for the successful reuse of requirements across system families that they contain sufficient information about the relationships between and context of requirements from early products in the family to assure developers that the requirements are valid for subsequent projects. Failure to observe this principle risks the late, and therefore costly, discovery of erroneous or unimplementable requirements.

In this paper, we offer a critique of standard traceability techniques and propose a method for developing traceability structures for requirements reuse.

2 Satisfaction Arguments and 'Rich Traceability'

Zave and Jackson [2] observe that satisfaction of a requirement (R) can be demonstrated only by a sufficient combination of domain knowledge (K) and specifications (S): S, K ⊢ R. Jackson suggests that traceability links between requirements and specifications should be supported by textual 'correctness arguments' which explain how the specifications and domain behaviour combine to provide assurance that the engineered system satisfies the requirement in the application domain [3]. Jackson's 'correctness' arguments have been integrated into several industrial-strength processes for requirements engineering [for example, 4]. The 'Rich Traceability' technique [5] represents 'satisfaction arguments' using goal-structures charting AND/OR decompositions, the justifications for which are recorded. Fig. 1 documents the 'rich traceability' relationship between a top-level operational requirement for a vehicle and lower-level technical requirements. The 'satisfaction argument' introduces domain knowledge and indicates that a conjunction of the three specifications satisfies the top-level requirement.

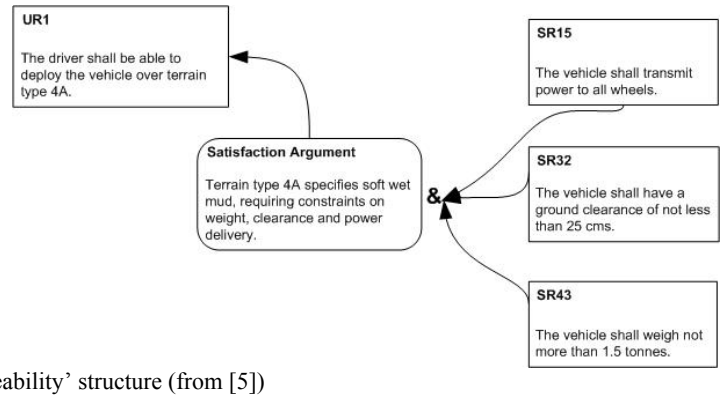

Fig. 1. A 'rich traceability' structure (from [5])

3 Locating the Satisfaction Argument

The inclusion of a 'satisfaction argument' in the 'rich traceability' structure (Fig.1) provides an explicit record of the strategy used to decompose requirements statements and permits tracking of relationships and constraints between requirements statements at different levels of abstraction. In this respect, 'rich traceability' offers considerable advantages over simple traceability structures, which simply postulate links between requirements artifacts, with no attempt to justify or explain the connections between them. We are concerned, however, that 'satisfaction arguments' and traceability structures of this type do not actually demonstrate the satisfaction of the top-level requirement by the lower-level statements.

Fig. 1 puts forward no argument to convince the reader that the refinement tactic adopted will result in a specification which satisfies the requirement. No evidence is presented to support the implied claim that power, clearance and weight are the only concerns which need to be considered in assessing the vehicle's capabilities in the terrain. Nor is there any argument to demonstrate that the technical constraints specified in the system requirements are relevant to the top-level requirement and sufficient to satisfy it. What is offered here is not a 'satisfaction argument', but a series of propositions which need to be supported by arguments to convince the reader that the design meets the requirement. The real 'satisfaction argument', then, is not to be found in the decomposition strategy recorded here, but in an unexpressed meta-argument stating the basis on which this strategy is proposed.

4 Satisfaction Arguments and Safety Cases

Previous work at York has focused on the development of a graphical notation method for the construction of safety case arguments, the Goal Structuring Notation (GSN) [6]. Safety cases are semi-formal arguments which demonstrate how available evidence about a system and its context can be used to show that a system is acceptably safe to operate in its context [7].

As applied in requirements engineering, GSN provides a means for recording traceability links between individual claims and sub-claims. These claims are represented as goals, and equate to the requirements statements and specifications in the 'rich traceability' example (Fig. 1). The notation also records the strategies used to decompose the goals. These strategies map to the 'satisfaction argument' in Fig. 1, in that they seek to provide a basis for the relationship between the goals. As well as presenting a clear record of the goal-decomposition strategy, however, GSN allows this strategy to be validated by the use of an apparatus of justifications and assumptions attached to goals and strategies within the structure, as well as by explicit references to artifacts such as system architectural models or contextual information. This documents the meta-argument capturing the satisfaction relationships which justify the decomposition of requirements. GSN expresses justifications and assumptions as propositions, and there is no opportunity provided for their further development. A more sophisticated treatment of the meta-argument can be achieved by the use of GSN 'away goals', by which reference is made to a parallel goal-structure in which the meta-argument (our 'real' satisfaction argument) is elaborated.

5 Meta-Arguments and Requirements Reuse for System Families

Satisfaction arguments, expressed as meta-arguments on a requirements decomposition, provide assurance that the traceability relationship between requirements at different levels of abstraction is valid within a given application domain. If requirements are to be reused successfully between projects, it is important that satisfaction relationships remain valid for requirements and specifications in the reuse domain. The following example demonstrates how GSN can be used to document satisfaction arguments, and to indicate where changes in design commitments or contexts challenge reused requirements.

Our example concerns a system family of full-authority digital engine controllers (FADECs) for civil airliners. A FADEC provides a computer-controlled management system for the engine, which takes inputs from the cockpit controls and sensors located on the aircraft and produces outputs in the form of digital signals used to control the engine [8]. Engines are developed as a system family, comprising

'marks' and 'variants'. A 'mark' is a specific engine in a series, such as the Rolls-Royce BR-710 and the BR-715, while a 'variant' is a mark produced to the specific requirements of an Airframer. Requirements for a particular 'mark' are expressed in terms of a 'common core', with variations associated with specific 'variants'.

Inadvertent deployment of reverse thrust while an aircraft is in flight can result in severe yawing and, at worst, loss of control. The common core requirements for a small-engine series (we'll call it 'Engine A') have four discrete check systems, all of which must be actively disengaged before reverse thrust is deployed:

1. An isolation valve must be opened, allowing the flow of hydraulic fluid into the thrust reverse system. This valve is controlled by the FADEC, and its default position is 'closed'.
2. The Thrust Directional Control Valve must be switched to 'reverse' from its default 'forward' position. This valve is the joint responsibility of the FADEC and the aircraft, although the aircraft (i.e. the pilot) has ultimate override control.
3. The mechanical Tertiary Locks holding the thrust reverser door in 'closed' position must be opened. The aircraft has responsibility for these locks.
4. Throttle interlocks provide the pilot with tactile feedback concerning the degree of reverse thrust, and prevent him from requesting more reverse thrust than he requires.

Fig. 2 records a simplified decomposition, in GSN, of the common core requirements for the thrust reverse deployment protection system in 'Engine A'. The decomposition strategy providing the traceability links between the checks (R2 - R5) and the top-level requirement (R1) is documented in the rhomboid labelled S1. A rich apparatus of contextual assumptions and definitions is recorded alongside the decomposition.

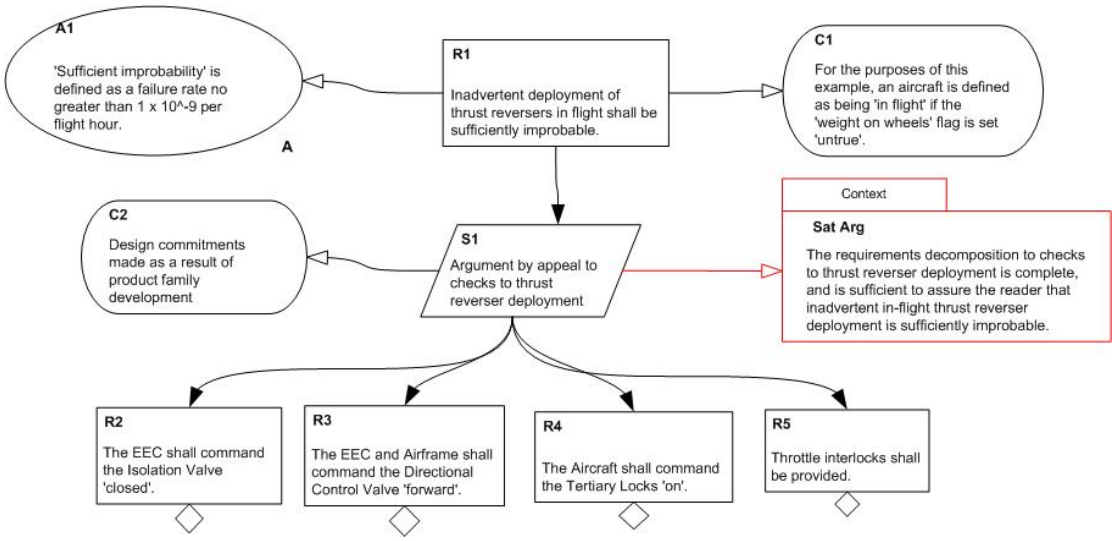

Fig. 2. Decomposition of core requirements for the thrust reverse deployment protection in the 'Engine A' family

The decomposition in Fig. 2 does not attempt to justify the decomposition strategy employed. Instead, it contains an 'away goal' reference to a justification claimed stored elsewhere (highlighted in red). This goal is the top-level claim of the meta-argument in Fig. 3, which is the satisfaction argument justifying the requirements decomposition. The argument strategy is a two-pronged one: the left-hand side of the goal structure argues that the checks, taken together, are sufficient to satisfy the top-level requirement, while the right-hand side (not fully developed here) argues that all possible failure modes have been considered and are adequately mitigated by the checks. The GSN structure makes clear what evidence is required to demonstrate the satisfaction of the top-level requirement (PSSA).

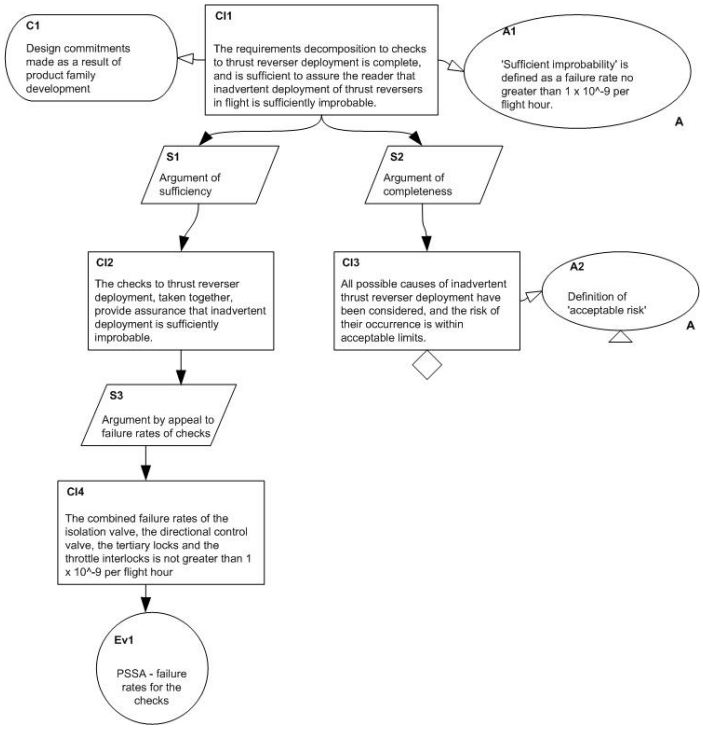

Fig. 3. Satisfaction argument for core thrust deployment protection requirements in the 'Engine A' family

For historical and regulatory reasons, Airframer customers for certain variants in the Engine A family do not require the throttle interlocks feature on their engines. Safe reuse of the core requirements with this modification depends on R1 being satisfied by the three remaining checks. The GSN structure provides a straightforward means for assessing which aspects of the satisfaction relationship are threatened by the design change. If R5 (throttle interlocks) is removed from the requirements decomposition, reference to the satisfaction argument demonstrates that the sufficiency argument is compromised. It will be necessary to consult the PSSA to see whether the combined failure rates for the isolation valve, the directional control valve and the tertiary locks can combine to satisfy the overall safety requirement of a failure rate not greater than 1 x 10^{-9} per flight hour.

GSN satisfaction arguments thus allow for the clear record of domain information and assumptions, and indicate which information sources are required for adequate requirements traceability. The severity of the impact of requirements or contextual change can be assessed by reference to meta-arguments on the requirements decomposition. This is of clear benefit in system family development environments, where requirements reuse depends on the assurance of the validity of requirements decompositions in alternative development contexts.

References

1. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. Proceedings of the 5th IEEE International Symposium on Requirements Engineering. IEEE CS Press, Toronto (2001) 249-263
2. Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, Vol. 6 No. 1 (1997) 1-30
3. Jackson, M.: Problem Frames: Analysing and Structuring Software Development Problems. Addison-Wesley, London (2001)
4. Hall, A.: A Unified Approach to Systems and Software Requirements. Proceedings of the 5th IEEE International Symposium on Requirements Engineering. IEEE CS Press, Toronto (2001) 267
5. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. Springer-Verlag, London (2002)
6. Wilson, S., Kirkham, P.: SAM User Manual. University of York, York (1995)
7. Kelly, T., McDermid, J.: Safety Case Construction and Reuse Using Patterns. Proceedings of the 16th International Conference on Computer Safety, Reliability and Security. Springer-Verlag, New York (1997) 55-69
8. Lam, W.: Achieving Requirements Reuse: a Domain-Specific Approach from Avionics. Journal of Systems and Software Vol. 38 (1997) 197-209