EVIDENCE-BASED DEVELOPMENT – COUPLING STRUCTURED ARGUMENTATION WITH REQUIREMENTS DEVELOPMENT

A.J.J. Dick*

**Integrate Systems Engineering, UK*

Keywords: progressive assurance, traceability, safety cases.

Abstract

This paper reports on an assurance process that is being applied on a large nuclear defence project in the UK. Christened Evidence-based Development (EbD), the approach draws together requirements development and structured argumentation into a unified assurance framework. Part of the discipline of requirements management is to document which design artefacts contribute to the satisfaction of which requirements by tracing individual statements of requirement through the layers of design. Evidence-based Development recognises that the systematic collection of "decomposition arguments" for each step in the requirements development, along with supporting evidence provided by design validation and verification activities, amounts to a structured argument for the design. The theoretical advantages of this approach are that assurance is coupled tightly to the design process, and that assurance is applied uniformly to all aspects of the design. However, it is too early in the life-cycle of the project to make definite claims for the approach. This far, the major barrier to the implementation of this approach has been the education of engineers across multiple disciplines in how to write the arguments entailed is each design step.

1 Introduction

Developers of complex systems are frequently required – by statute, regulation, standards and customers – to provide evidence that their products are fit for purpose. Even before a product is built, developers may be required to supply design justifications, compliance statements or safety cases for review and sign-off as part of the development process. While the system is being built and tested, it may be necessary to gather and present further compliance evidence. And while in operation, evidence for the correct functioning of the system may be required. The effect is to accumulate a growing body of evidence for the correctness, compliance and safety of a system throughout its development life-cycle.

The concern with accumulating evidence is particularly relevant to the challenges involved in validating high integrity systems and hence achieving certification against standards such as DO-178B for avionics or obtaining Premarket Approval (PMA) for medical devices. To meet these challenges, frameworks, processes and tools are required that help coordinate and organise the collation, review and publication of validation and certification evidence.

System safety is one area in which techniques, notations and tools have been deployed for the construction of safety cases. However, safety has to be balanced with other concerns – such as delivering effective capability – and unless a commensurate degree of rigour is applied to these other areas, safety can have an undue influence on the nature of the design.

Evidence-based Development (EbD) is an approach to systems and software development that provides a uniform framework for structured arguments across all pertinent aspects: function, performance, safety, reliability, etc. It couples the assurance case tightly to the design process, starting with the way in which requirements are developed.

EbD is been implemented on a large UK defence project. The major emphasis has been on getting the process in place with appropriate tool support using IBM Rational DOORS, and then training and mentoring the users.

2 Rich traceability

Much of the discipline of requirements management focuses around traceability: the ability to trace requirements as they are transformed from customer needs to design specifications through the layers of design. Effective tracing depends on requirements being expressed as concise, singular, unambiguous statements appropriate to their level of abstraction. (See, for example, the guidelines in ISO/IEC 29148 [1].)

Rich traceability [2, 3] introduces the notion of rationale for the decomposition of a requirement that we call "decomposition argument" in the paper. In the rich traceability literature, this is usually called a "satisfaction argument"; however, if one considers how one may argue that a requirement is satisfied, one must take into account how the requirement is tested as well as how it is decomposed. We therefore reserve the term "satisfaction argument" for something broader that decomposition. Note also that the decomposition argument is different from the notion of rationale for an individual requirement, which explains its

existence; the decomposition argument explains the decomposition of a requirement into one of more others, and justifies the design.

Figure 1 shows an example of a system requirement being decomposed into a number of component requirements along with the decomposition argument.

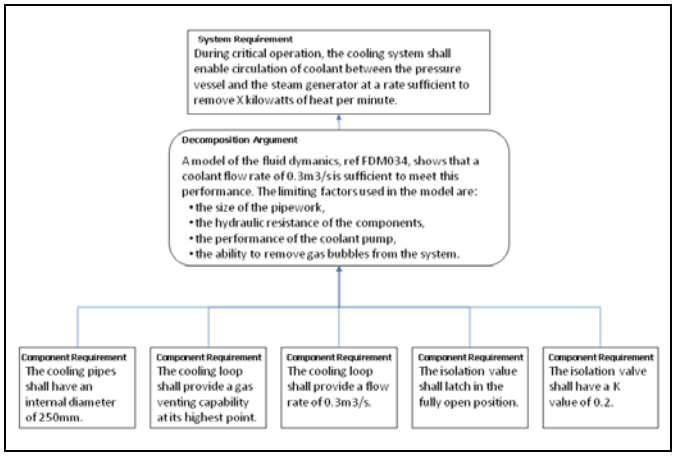

Figure 1: Example of rich traceability for the decomposition relationship

The decomposition argument should address two key things:

- *Sufficiency:* Why is the set of child requirements sufficient to satisfy the parent?
- *Necessity:* Why is each of the child requirements necessary to satisfy the parent?

These two concerns give criteria for reviewing the relationship between layers of requirements.

Decomposition is not the only traceability relationship that exists as part of the discipline of requirements management. Another is the relationship between requirements and V&V information, which we call the "qualification" relationship. The principle is to trace (planned) validation or verification actions to the requirements that they are intended to establish.

The concept of rich traceability can be extended to the qualification relationship. A "qualification argument" is captured against each requirement that explains the selection of V&V actions. Figure 2 shows an example.

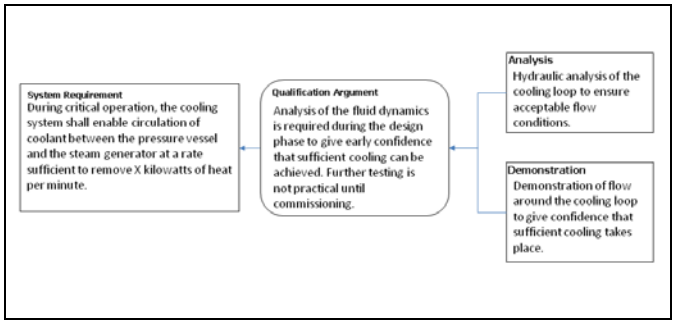

Figure 2: Example of rich traceability for the qualification relationship

Sufficiency and necessity also apply as criteria for writing a qualification argument: Why is the set of V&V actions sufficient to show the requirement has been met? and Why are each of the actions necessary? (In the testing world, sufficiency is often called adequacy.)

2 The Semantics of Traceability

Associated with each traced relationship, such as those shown above, is an implicit claim. In the case of Figure 1, the claim is that, if the functional requirements are satisfied, then so is the user requirement. For Figure 2, the claim is that, if the planned tests show positive results, then the requirement has been met.

This observation allows us to connect rich traceability to the concept of claim/evidence/argument [4, 5]. The systematic collection of decomposition and qualification arguments provides an overall structured argument for the satisfaction of requirements, structured according to the way in which requirements are decomposed, which in turn reflects the design.

The structured argumentation consists of implicit claims, explicit arguments, and evidence that comes from the execution of V&V actions. Every V&V action is, in effect, a request for evidence. The V&V actions include those that occur very early in the life-cycle, such as design analysis, those that occur repeated at various stages, such as hazard analysis, right through to those that occur far later, such as component tests, integrate tests, system tests and acceptance tests.

Decomposition and qualification arguments, which are made at the time of design, reflect the *intent* of the design, since nothing has yet been built. The decomposition argument is supported by V&V actions involving various kinds of design analysis that also consider the *intent* of the design. Later in the life-cycle, when components have been built and integrated, the V&V actions that test them and the evidence they generate, are concerned with the *fulfilment* of the design. Thus the overall argument structure evolves over time from considering intent to considering fulfilment.

4 Evidence-based Development

Evidence-based Development (EbD) is a process based on the above principles. Rich traceability is applied through the decomposition and qualification relationships at every level of abstraction, with arguments being collected at each design step. As development proceeds, so the structure of the overall argument develops and evidence accumulates, providing – if all goes well – increasing confidence in that the system is fit purpose against all kinds of requirement.

Figure 3 presents an EbD framework based on an extension of the V-model into a W-model. The purpose of the extra axis is to make the test planning steps explicit in the development

process. The rounded boxes are activities, the rectangles information, and the arrows traceability relationships.

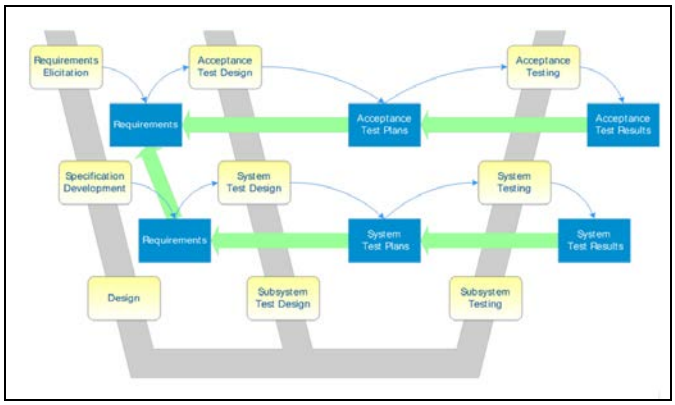

Figure 3: The W-model in which test plans are made explicit

Figure 4 overlays the W-model with a trace of decomposition and qualification relationships leading from a single top-level requirement. The small rectangles are requirements and tests, and the circles are decomposition and qualification arguments. This structure represents the complete assurance case for the one top-level requirement.

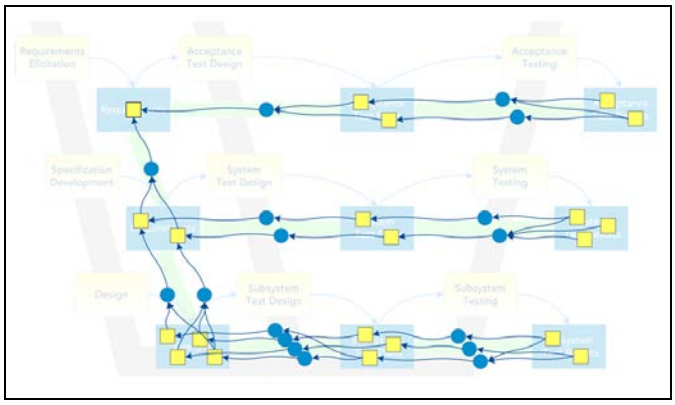

Figure 4: Example of rich traceability for the decomposition relationship

Since EbD applies the same processes to every requirement, assurance is established in a uniform and balanced way across the range of different types of requirement.

5 Implementation

Here we describe the way in which EbD has been implemented in a large defence project. The EbD process and information is defined at two levels: a high-level model defines the layers of requirements that are to be managed, and the requirement areas in those layers. The low-level model describes the statement-level relationships and processes that apply within an area. Exactly the same low-level-model is instantiated in every area in the high-level model.

Figure 5 shows the high-level model for the project in question. The yellow boxes show requirement areas and the red arrows show how the requirements are allocated from one area to another. Lines draw across the model show requirements layers. Transverse requirements include types of requirement that cut across the product structure, such as safety, availability and maintainability.

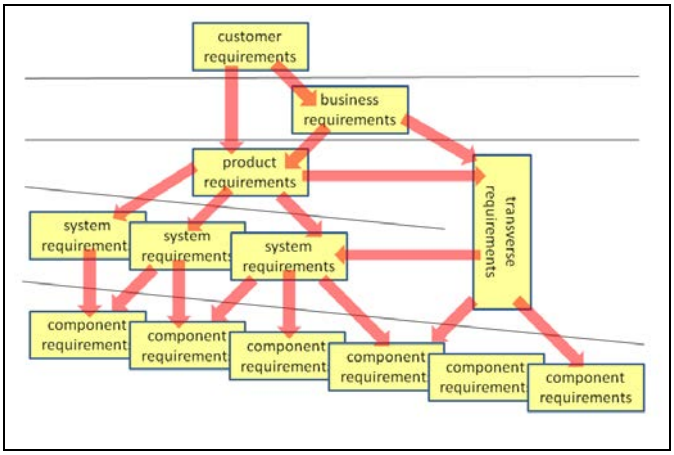

Figure 5: High-level model showing layers of requirements

Figure 6 shows the low-level model that is applied in every requirements area. It portrays the two key relationships (decomposition and qualification), along with four kinds argument that are collected, described as follows:

1. *Decomposition argument:* rationale for why the requirement is decomposed/allocated in the way that it is. (An attribute of the requirement.)
2. *Qualification argument:* rationale for why the particular V&V methods for this requirement have been selected. (An attribute of the requirement.)
3. *Qualification results argument:* rationale for why the test results show that the criteria for each V&V method have been met. (An attribute of the V&V method.)
4. *Satisfaction summary:* overall rationale for believing that the requirement has been satisfied. (An attribute of the requirement.)

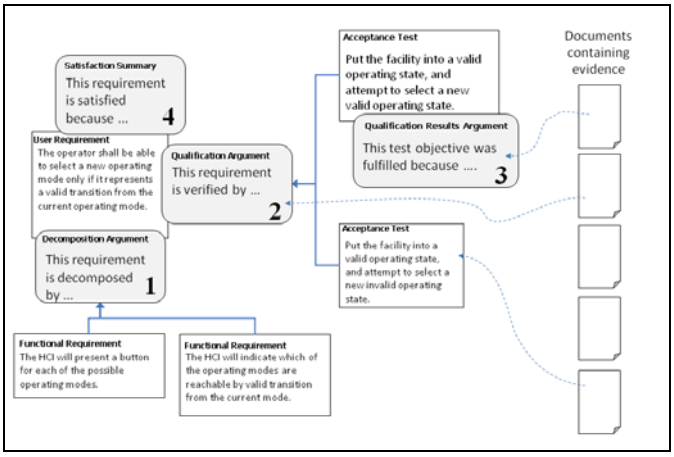

Figure 6: Low-level model showing 4 kinds of argument

The area owners collect the arguments as they develop the requirements, plan V&V actions against them and execute

those actions. The order in which the arguments are captured is usually as shown. The last argument collected is the satisfaction summary, which draws together all the arguments and evidence into a single overall statement of satisfaction.

As mentioned before, the decomposition and qualification arguments are concerned with the intent of the design, whereas the qualification results and satisfaction arguments are concerned with fulfilment of the design.

Evidence supporting the arguments lives in documents outside of the requirements database. Some of these documents may be produced by V&V actions that generate evidence to support the decomposition and qualification arguments stated against the requirement.

Tool support for the EbD process in provided through a customised DOORS database.

6 Implementation Challenges

The application EbD to the development organisation required a considerable change in culture. The 240 engineers trained in the process and the supporting tools were not accustomed to the discipline of writing concise, singular, unambiguous requirement statements, let alone to the discipline of systematic tracing with rationale. Therefore, there was a great deal for the practitioners to take on board at the same time.

The approach used to promote the culture change was to extend class-room training into at-the-desk mentoring, thus helping engineers through the steps of the process as they faced day-to-day challenges.

What became quickly apparent was that writing arguments is a very different discipline to writing requirement statements. Particular challenges in writing decomposition arguments are:

- Not simply repeating the child requirement statements.
- Not reproducing the whole design; just summarising the pertinent details.

Guidelines for writing arguments are lacking in the project. Another challenge has been confusion between the roles of 4 different types of argument collected.

Since the project is still in the design stages, not many of the Results and Satisfaction arguments have been captured. It will take years before the complete argument structure is in place, and the quality and nature of the resulting assurance cases can be fully assessed.

7 Lessons Learned

Some of the challenges mentioned above could be addressed by applying a simplified approach to the collection of arguments. We observe that the qualification argument was often not independent of the decomposition argument, in that, when deciding what V&V actions to place against a requirement, you have to take into account how that requirement is decomposed and what V&V actions are placed against the children.

We also observe that decomposition and qualification arguments sometimes evolve as understanding of the design improves. Indeed, with the length of the overall programme, we find we want to write the overall satisfaction argument early to reflect current assessment of the requirement, and let that argument evolve as more evidence becomes available from the results of V&V actions.

These observations lead to the idea of having a single, evolving argument against each requirement – the satisfaction argument – that takes into account the concerns of decomposition, qualification and associated evidence, and simply reflects the current rationale for believing the requirement will be, or has been, satisfied.

The multi-argument approach encourages the developers to collect assurance information in a systematic was as the development of the design proceeds, and provides an explicit record of how the overall argument has developed through the stages of development from design intent to design fulfilment. In contrast, the single argument approach would record only the latest state of the argument. Configuration management could be used to trace the way the argument develops. It is unclear at this stage whether this single argument approach would produce the same quality of structured argument as the multi-argument approach.

7 Conclusions

The aim has been to present a potentially interesting approach to the systematic collection of assurance information tightly coupled to the development of the design. It is too early to be able to draw any firm conclusions about the nature and quality of the assurance case thus produced.

References

[1] ISO/IEC, *ISO/IEC 29148 FDIS Systems and Software Engineering—Life Cycle Processes—Requirements Engineering, 2011*.

[2] "Rich Traceability", Jeremy Dick, Proc. 1st International Workshop on Requirements Traceability, Edinburgh, Sept 2002

[3] "Requirements Engineering", M. Elizabeth Hull, Ken Jackson, A. Jeremy J. Dick, Springer-Verlag, London, Sept 2002, ISBN 1-85233-577-7

[4] T.P. Kelly and R.A. Weaver, “The Goal Structuring Notation—A Safety Argument Notation,” Proc. Workshop Assurance Cases Dependable Systems and Networks, 2004

[5] R E Bloomfield, P G Bishop, C C M Jones, P K D Froome, ASCAD—Adelard Safety Case Development Manual, Adelard 1998, ISBN 0 9533771 0 5.