

Evidence-based Development

How to reduce risk, improve quality and ensure compliance

An Integrate white paper

May 2009

Evidence-based Development: how to reduce risk, improve quality and ensure compliance

Abstract

This paper presents Evidence-based Development (EbD), a confidence-based approach to progressive system assurance that is closely integrated with the development process.

EbD draws inspiration from requirements management, from risk management, and from the claim-evidence-argument paradigm – well known in the safety engineering domain – whose ideas we extend from safety to the broader concept of system fitness-for-purpose.

EbD provides an evidential backbone for assurance. It establishes assurance as a progressive activity starting from the very outset of the system lifecycle. Evidence is accumulated where confidence is most lacking, beginning with design verification in the earliest stages of development, through to design fulfilment from test results in the later stages. It caters for assurance and compliance evidence arising in many different forms from diverse sources and approaches throughout the lifecycle.

As an example of its application, the paper summarises how EbD can be applied to the certification of systems to RTCA/DO-178B. It argues that EbD has features that address particular parts of this standard in new ways.

By making assurance an integral part of system development, EbD promises a range of benefits, including reduction in life-cycle costs and risks for systems in which a major concern is high integrity or compliance to regulation.

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. Evidence-based Development (EbD[®]) aims to meet to this challenge.

System safety is one area in which techniques, notations and tools have been deployed for the construction of safety cases. EbD differs in two key respects: it applies the same degree of rigour to concerns other than safety, and it strongly connects the development of an assurance case with the development of the system itself.

EbD is an approach to systems and software development that delivers progressively increasing confidence in the validity of development artefacts. As confidence increases, risk is reduced and quality becomes assured. The essence of EbD is to capture a growing body of evidence for the correctness of the system in all pertinent aspects: function, performance, safety, reliability, etc. The progressive assurance starts from the earliest possible stages with rationale associated with the verification of design against requirements, through to the fulfilment of the design as evidenced by the associated test results. It is a uniform approach to argued

verification and validation that is fully integrated with the development process, ensuring that the assurance case is developed in parallel with development. In this way, system assurance is not a separate activity carried out by a separate team, but is performed by the developers (who are the best informed) and brings immediate benefit to the developers in terms of better understanding, greater confidence and improved ability to manage change.

Assurance Cases

The core objectives of system assurance are to build confidence and to reduce risk. Assurance is typically concerned with whether:

- emergent technical solutions will satisfy their requirements;
- these solutions are derived from controlled, appropriate and effective processes;
- deployed systems will be fit for purpose.

To this end, assurance is interested in answers to the following kinds of question:

- Are our requirements complete and correct? (requirements validation)
- Do our designs discharge our requirements? (design verification)
- Do implemented systems comply with designs? (system verification)
- Are our processes appropriate and effective? (design assurance)
- Are deployed systems fit for purpose? (system validation and certification)

Some assurance concerns can be addressed very early in the development lifecycle, even before an end product has been built. For instance, as soon as stakeholder (customer or user) requirements exist, validation and verification planning can begin, and evidence relating to the selected verification techniques can be collected.

The emphasis of assurance changes throughout the lifecycle. Initially, the focus is on design intent; in other words, arguments are constructed relating to the intent of the design as expressed in terms of requirements and specifications. This enables an assurance case to be developed and reviewed early in the life-cycle, building confidence in the design approach. The kind of evidence collected at this stage is typically based on analysis, modelling and analogy. Later in the life-cycle, when systems have been built and testing can begin, the focus is on design fulfilment, building confidence that the requirements have been met. Here the kind of evidence collected typically comes from

measured test results and inspections, supplemented as necessary by further analysis to argue the case for acceptance in the light of these test results.

Requirements Management

The ability to construct an assurance case depends crucially on how the artefacts of development are managed and traced. Development artefacts in this discussion include statements of requirement, specifications, tests and designs. As development proceeds, artefacts are created, and the relationships between them are traced.

For the purposes of illustration, we review here some fundamentals of requirements management that address these issues. Two topics are covered: expressing requirements and tracing requirements.

Expressing Requirements

Requirements management is the discipline of eliciting, expressing, satisfying, verifying, tracing, evolving and reusing requirements. In managing requirements (and other development artefacts), the significant focus is on the individual statements of requirement and the processes they engage in rather than on the documents that contain them. In particular, the ability to trace requirements depends first on being able to write traceable statements. For this reason, the way in which those requirements are expressed is crucial.

Here are some qualities that are typically cited as desirable in the way a statement of requirement is expressed:

- *singular*: each statement is a single traceable element;
- *identified*: each statement is uniquely identified;
- *understandable*: each statement is clear and precise;
- *unbiased*: does not impose a solution on the next layer;
- *quantified*: each statement has acceptance criteria;
- *testable*: each statement can be validated / verified;
- *traced*: to satisfying requirements and qualifying tests.

Expressing requirements in this way allows each to be managed, and development processes – including assurance – to be strongly connected to the principal objectives of the product.

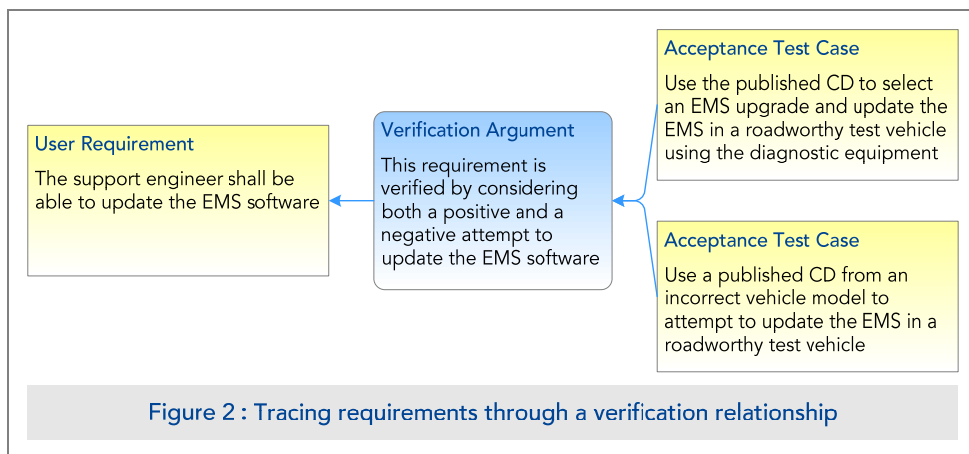
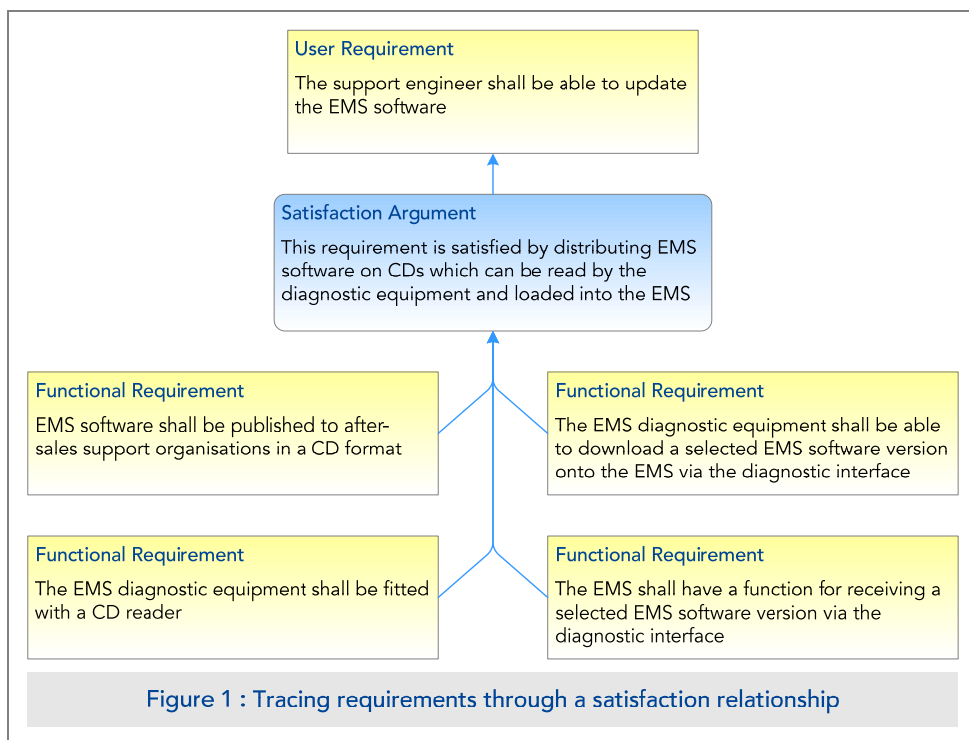
Tracing Requirements

Requirements tracing is a means of recording relationships between development artefacts to enable various forms of analysis, including impact assessment. There are many kinds of relationship that are useful to capture, such as the satisfaction of one layer of requirements by another, the verification relationship between planned tests and requirements, relationships between risks and other artefacts, and so on.

Figure 1 shows an example of tracing a user requirement through the satisfaction relationship to four system-level functional requirements. Figure 2 shows tracing through the verification relationship to show which acceptance tests are associated with the same requirement.

Artefacts should be traced as they are developed, because that is when the developers know why the relationship is pertinent. In this way, creating the tracing is a very small overhead on the development process, and the benefits outweigh the extra work.

The benefits of tracing are immediate as well as longer term. In the immediate, the discipline of capturing tracing and the associated arguments engenders greater reflection on the part of the developer, in turn improving the design and confidence in its validity. In the longer term, the record of the relationships and rationale will pay dividends in assessing the impact of change when external or internal pressures cause the design to evolve, and will provide the evidence to underpin an assurance or compliance case.



Rich 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 we can be confident that the requirement has been met.

Rich traceability articulates this implicit claim in the form of an explicit argument. The nature of the argument depends on the relationship and on the stage of development.

Figure 1 shows the satisfaction argument as a rounded box on the relationship. Since the satisfaction argument is not a statement of requirement, it is expressed in a different style of language. In this case, it starts with the phrase "This requirement is satisfied by ...".

Figure 2 shows a verification argument associated with the traced relationships from the requirements to the planned tests. Here the argument begins with the words "This requirement is verified by ...".

An additional immediate benefit of capturing these arguments is that it exposes the design rationale to peer review. Not only can the requirements be reviewed at various levels, but the relationships between them as well. When reviewing a satisfaction argument, there are two key considerations relative to the implicit claim:

- *Sufficiency*: is the set of lower requirements sufficient to satisfy the top one?
- *Necessity*: is each of the lower requirements necessary to satisfy the top one?

Likewise, when reviewing verification arguments, consider the corresponding considerations:

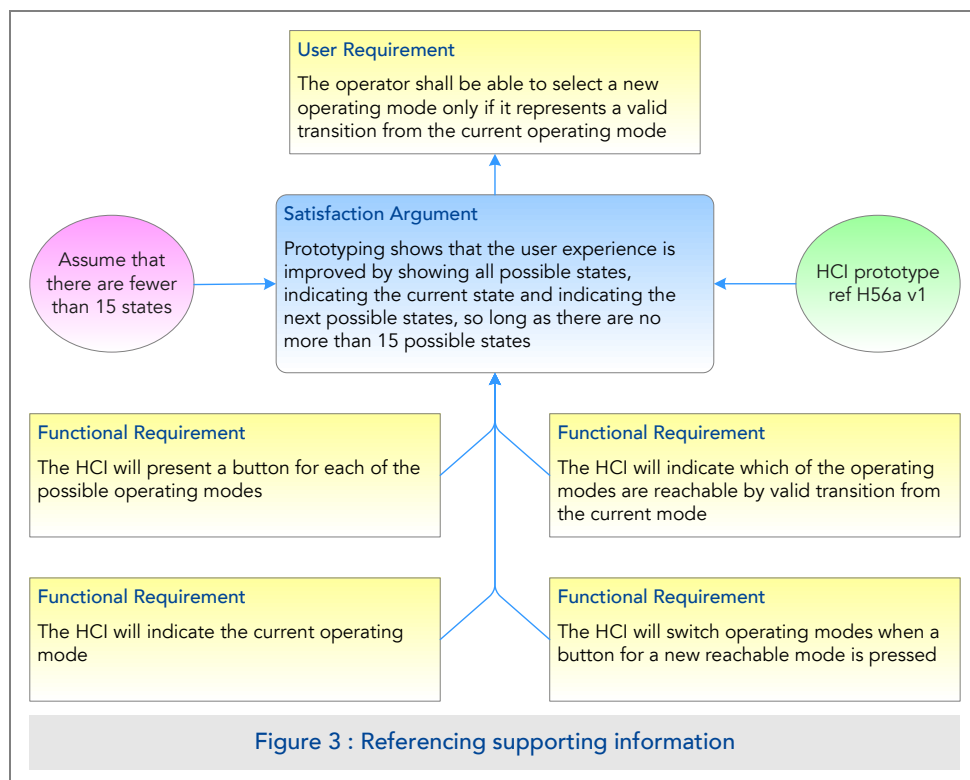
- *Adequacy*: is the set of planned tests adequate to verify the requirement?
- *Necessity*: is each of the planned tests necessary to verify the requirement?

Supporting Information

The satisfaction, verification or other arguments are also the hub against which supporting information can be collected. Arguments are supported by evidence, assumptions and other contextual information. Figure 3 shows another example of a satisfaction argument with reference to an assumption and a prototyping activity.

When reviewing supporting information, the following should be considered:

- *Appropriateness*: is the evidence being presented appropriate to the kind of argument presented?
- *Relevance*: is the evidence pertinent to the argument?

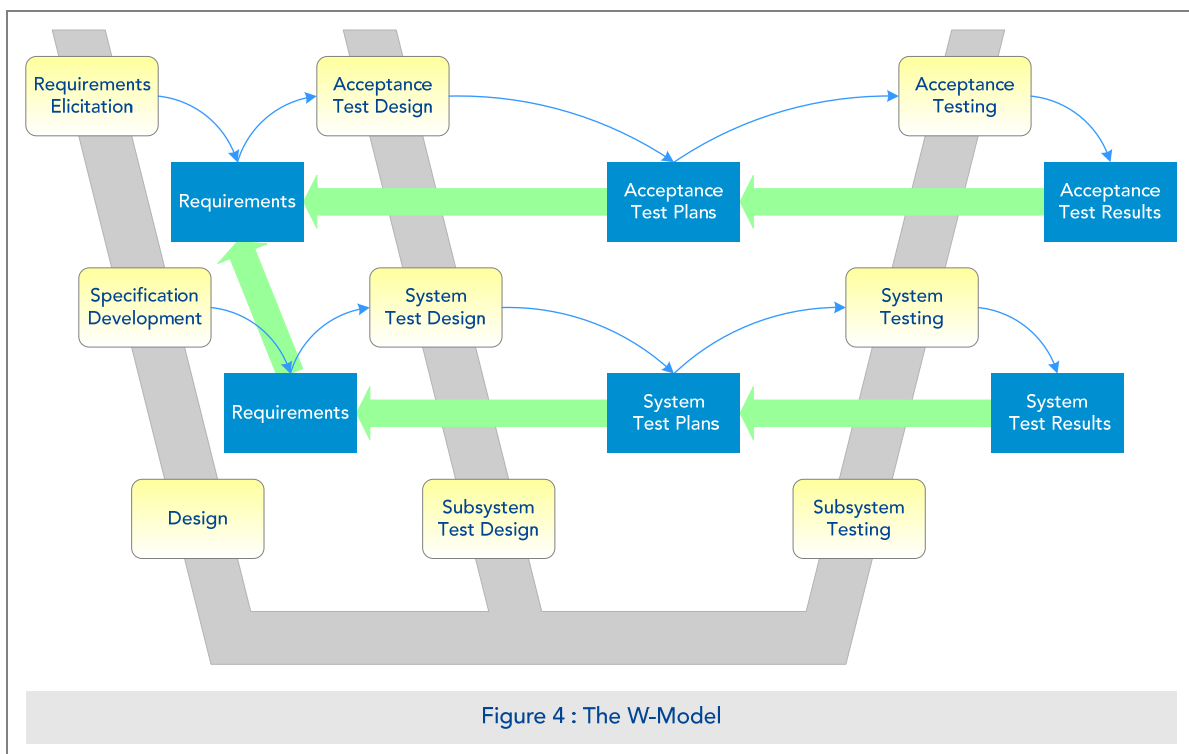


The W-Model

To illustrate further the shifting focus of assurance, consider an extension of the classic “V-model” shown in Figure 4, referred to as the “W-model”. An axis is added that covers the planning and design of tests against requirements specifications and designs. The rounded boxes are activities, the rectangles represent products of those activities, and the thick arrows represent tracing between artefacts, i.e. which specification components satisfy which requirements, which tests cover which requirements, etc.

The three axes represent significant phases of development:

- *Analysis phase:* elicitation and development of requirements and validation of requirements with customer;
- *Qualification planning phase:* planning of tests, assessments, analysis and other qualification activities to demonstrate requirements achievement;
- *Qualification phase:* collation and analysis of qualification evidence from test results and other activities.



Associated with the axes are the thick green arrows that represent tracing relationships. Using rich traceability against each artefact, these relationships are the focus for three phases of assurance, summarised in Table 1.

Through consistent application of rich traceability during these phases, an assurance case is produced progressively, giving growing confidence in the system being developed, thus inherently reducing risk and improving quality. Figure 5 shows a complete assurance case associated with an individual user requirement – the one shown in the top left corner. As the system is designed, the arguments for satisfaction and verification are developed in parallel down the left-hand side; then, as the system is built and tested, the fulfilment arguments are provided. The complete case for the requirement finally consists of the connected set of artefacts, arguments and referenced evidence.

Practical Concerns

Developing arguments and collecting supporting evidence requires time and expertise. The unconsidered application of an approach such as EbD against every development artefact may raise costs above the level of benefit. Here we consider two aspects that address these concerns.

Establishing Confidence

One approach to determining which artefacts warrant the additional effort required to develop arguments and collect evidence is to make a confidence assessment against each claim associated with an artefact. For instance, requirements differ widely in terms of their criticality and the level of assurance risk they present. Some may be in familiar territory for the discipline or development team, while others are novel or known to be problematic. EbD is then

applied systematically in measures commensurate with the need to raise confidence.

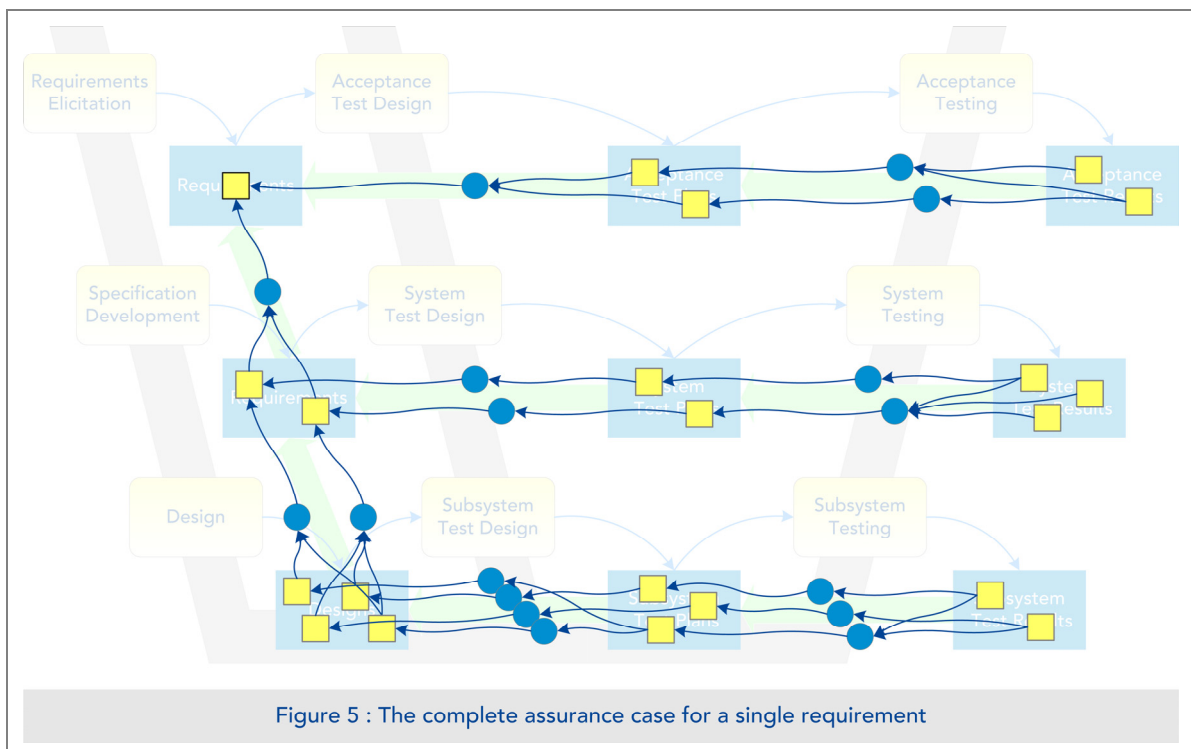
Much the same approach is applied in risk management. Risks are assessed for inherent magnitude, and controls put in place to reduce

the likelihood or impact of a risk where necessary. Mitigations are assessed for residual risk.

Using this approach, EbD is focused on raising confidence rather than on absolute assurance, and effort is expended where most benefit will be accrued.

Stage	Claim	Argument	Evidence
Design intent	Achievement of supporting requirements will cause this requirement to be satisfied	Satisfaction argument about the sufficiency and necessity of the design	Traceability to lower level requirements References to analysis and modelling (design, cost-benefit analyses, etc)
Qualification intent	Planned qualification will deliver sufficient confidence that the requirement has been achieved	Verification argument about the adequacy and necessity of the planned tests	Traceability to supporting tests References to analysis and modelling (test coverage, test adequacy, etc)
Fulfilment	Result of qualification activities gives sufficient confidence that the requirement has been achieved	Verification argument about why the test results show that the requirement has been met	Test and qualification results References to test result analyses

Table 1 : Stages of assurance



Reusing Argument Structures

When repeatedly developing systems in a particular domain, the same argument structures will reappear again and again. A promising way of reducing the cost of creating an assurance case is to reuse the structures. Many of the same issues arise here as when reusing software: a very careful review of the reused claims and arguments has to be carried out, and much of the evidence collected afresh in the new context.

Evidence-based Development

Evidence-based Development (EbD[®]) is the systematic application the claim-argument-evidence approach across the W-model. Based on rich traceability, it is strongly connected to the development process, provides the evidential backbone for the system development process, and allows a comprehensive assurance and

compliance case to be created in parallel with other engineering activities.

A Micro-process

EbD is, in effect, a micro-process that applies everywhere a development artefact engages in a relationship (satisfaction, verification, or other) that requires an argument. The process covers the creation of the artefact, followed in parallel by the development of the structures relating to the relationship, the formulation of the argument, collection of the evidence, and identification of other supporting information such as issues. Finally, the artefact and its associated information is reviewed and approved.

Tool Support for EbD

Effective tool support is essential for applying EbD. Figure 6 shows an example of such a tool: Integrate’s TraceLine™ for DOORS.

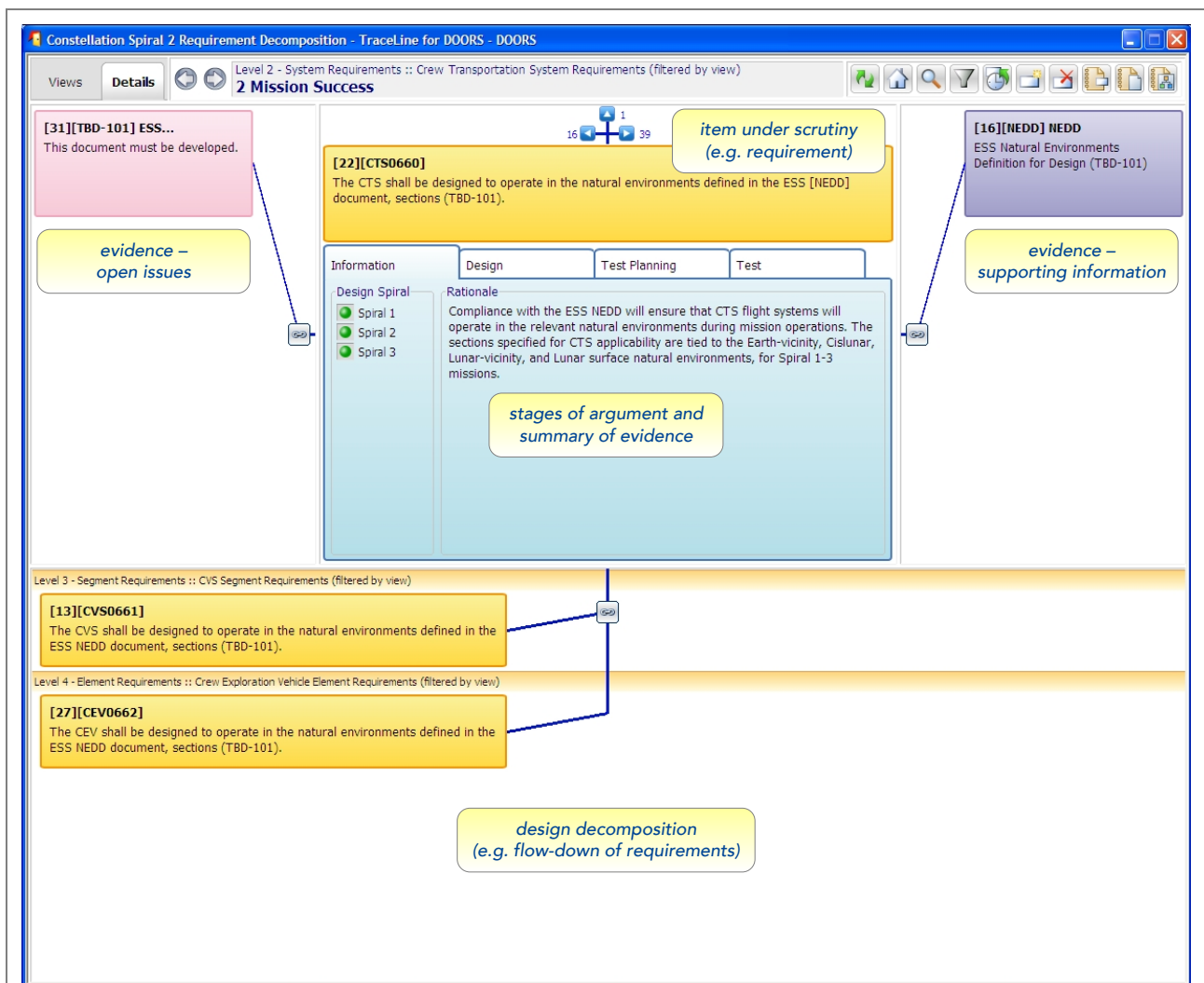


Figure 6 : Screenshot of TraceLine™ for DOORS

TraceLine is an extension to IBM Rational® DOORS®. DOORS is a leading requirements management application that can help you reduce costs, increase efficiency and improve quality by enabling you to optimise requirements communication, collaboration and verification – throughout your organisation and across your supply chain. It can help you capture, link, trace, analyse and manage changes to information. So it's easier to make sure that your projects meet business goals, satisfy customer needs, and address applicable regulations and standards. Rational DOORS has been used to manage some of the world's largest projects that in some cases have included millions of requirements and hundreds of thousands of traceability links – from business objectives to component requirements.

TraceLine provides DOORS users with a powerful and intuitive graphical browser that lets you view and manage your DOORS information in a rich, visual environment. You have uncluttered access to the complete set of information you need because TraceLine hides information that isn't relevant to your current task. With TraceLine, you increase user productivity, reduce training needs, and provide non-technical staff with easy access to your DOORS information.

TraceLine is easily configured by the user to support EbD. The screenshot in Figure 6 presents information about a single development artefact as follows:

- Centre-top shows the item under scrutiny such as a requirement statement or test definition;
- In the middle are the arguments, tabbed for each stage;
- The lower panel lists the traced items, such as contributing requirements or test plans;
- The side panels show supporting information, such as assumptions, issues, and references to analysis and modelling.

The tool allows the user systematically to gather information around the artefact, to decompose the requirement with supporting satisfaction argument and evidence, to plan tests against the artefact with supporting verification argument and evidence, and to navigate through the information model to apply the same process to other artefacts.

Significant is the ability to publish from this tool the complete assurance case for a particular layer of development. This produces a hyperlinked report listing all of the claims, arguments and supporting evidence, which can be published electronically or on paper for review and sign-off purposes.

An example: Evidence-based Development and DO-178B

DO-178B, "Software Considerations in Airborne Systems and Equipment Certification", provides guidelines for the production of software for airborne systems and equipment. It aims to assist in developing software that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. The standard lists objectives that should be attained for 4 software levels based on consequence of failure, and contains descriptions of the evidence that should be collected to show that the objectives have been satisfied.

The layers of requirements named in DO-178B can be expressed in terms of the W-model as illustrated in Figure 7, where the stages of argumentation are also shown. In particular, there are two stages of verification intent, one between the requirements and the test case documents, and another between the test cases and test procedures.

While EbD applies to systems rather than just software, it supplements DO-178B by enforcing the collection, review and presentation of evidence of intent associated with the relationships between these layers of requirements. EbD reinforces the following DO-178B software development and software verification steps, and provides evidence that these steps have been performed:

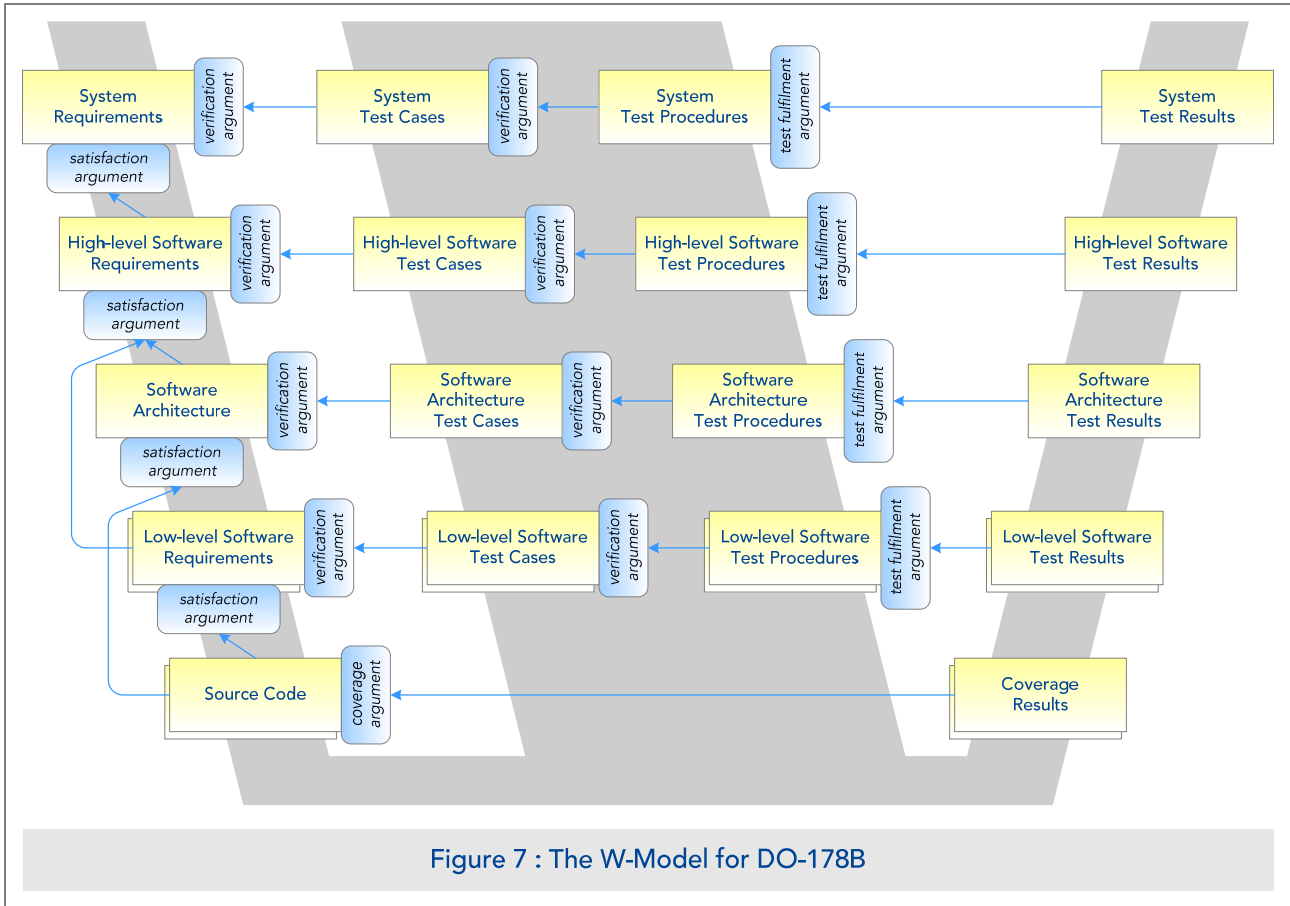
- the system requirements analysis process;
- the system requirements clarification process;
- traceability between software high-level requirements and system requirements;
- the derivation of requirements from hazard analysis;
- traceability of software requirements to system requirements;
- that every system requirement allocated to software is satisfied by one or more high-level software requirements;
- that every high-level software requirement has been developed through the architecture into one or more low-level requirements that satisfy it.

EbD also addresses the collection of evidence for verification, both at the planning stage (intent) and assessment stage (fulfilment). The following DO-178B steps are concerned:

- that every requirement is covered by appropriate verification actions (requirements-based coverage analysis); and

- that those requirements that cannot be tested have other justified test actions performed on them.

This suggests that EbD is a beneficial framework for supporting DO-178B in general, and has features that address particular parts of the standard in new ways.



Conclusion

EbD comprises information structures and processes for the collection, organisation, review, change and presentation of assurance cases to support certification and acceptance of complex systems. It also provides guidance to allow practitioners to judge the depth of application of the approach that brings the most cost benefit in a given context.

The benefit of applying EbD is that confidence in the eventual validation and certification of the system builds steadily from the beginning of the life cycle and that, when certification or acceptance is sought, a body of well structured and appropriate evidence is easily available. Because the arguments and evidence are so closely connected to the development activities and artefacts, assurance is also integrated with change management: impact analysis based on tracing also reveals the arguments that need to evolve, and the effect of a proposed change in the assurance case.

Systematic adoption of EbD has the potential significantly to reduce the life-cycle costs of high

integrity products and systems by reducing risk, improving quality and improving the integrity of the certification and acceptance process.



TraceLine for DOORS is validated as Ready for IBM Rational software

About the author

Dr. Jeremy Dick was educated at London University's Imperial College in Computing Science, majoring in formal methods of software development. He went on to pursue academic and industrial research in this field for a number of years. From 1996 to 2006, he worked for software tool supplier Telelogic (now IBM) in their UK professional services organization, as a principal consultant in tool-supported requirements management. This role has afforded him a broad exposure to requirements management practices and issues across many industry sectors. In this role, he developed the concept of Rich Traceability. He co-authored the Springer book entitled "Requirements Engineering," and toured internationally giving seminars in this subject. In 2006, he moved to UK-based consultancy Integrate Systems Engineering Ltd, where he has been helping to develop and promote the concept of Evidence-based Development. He is a past Chairman of INCOSE's Requirements Working Group.

About Integrate

integrate's vision is for companies to align strategy and execution, drive quality and ensure compliance through using systems engineering techniques as the bridge between the domains of engineering and management. Our consultancy practice and supporting technologies help clients deliver this vision. We use systems engineering as a vehicle to support informed management and direction, and to optimise solutions against both technical and business constraints. Our approach is intensely pragmatic, firmly focused on our clients' delivery needs, and founded on a thorough understanding of the underlying problem.

For more information go to www.integrate.biz or email info@integrate.biz

EbD®, TraceLine™ and Evidence-based Development™ and are trademarks or registered trademarks of Integrate Systems Engineering Ltd.

The Ready for IBM Rational Software mark and the trademarks contained therein are trademarks of IBM Corp. IBM is not the licensor of this Business Partner's product and does not make any warranties regarding this Business Partner's product.

IBM, the IBM logo, ibm.com, Rational and DOORS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml. Other company, product, or service names may be trademarks or service marks of others. References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided "as is" without warranty of any kind, express or implied. Any performance data for IBM and non-IBM products and services contained in this document was derived under specific operating and environmental conditions. The actual results obtained by any party implementing such products or services will depend on a large number of factors specific to such party's operating environment and may vary significantly. IBM makes no representation that these results can be expected or obtained in any implementation of any such products or services.

Any material included in this document with regard to third parties is based on information obtained from such parties. No effort has been made to independently verify the accuracy of the information. This document does not constitute an expressed or implied recommendation or endorsement by IBM of any third-party product or service.

Find out more

If you require further information, please contact

David Guy

t: +44 (0)1225 859991

m: +44 (0)7855 379022

e: david.guy@integrate.biz

Trefor Hooker

t: +44 (0)1225 859991

m: +44 (0)7837 526247

e: trefor.hooker@integrate.biz

integrate systems engineering ltd

251-253 London Road East

Batheaston

Bath BA1 7RL

United Kingdom

www.integrate.biz

© 2009 integrate systems engineering ltd

All rights reserved. This document is copyright of Integrate Systems Engineering Ltd.

Integrate Systems Engineering Ltd is registered in England no 4588165 and has its registered office at 21 St Thomas Street, Bristol BS1 6JS