The Indispensable Role Of Rationale In Safety Standards

John C. Knight and Jonathan Rowanhill

Dependable Computing, Charlottesville, VA, USA

{john.knight, jonathan.rowanhill}@dependablecomputing.com

Abstract. In this paper, we argue that standards, especially those intended to support critical applications, should define explicitly both the properties expected to accrue from use of the standard and an explicit rationale that justifies the contents of the standard. Current standards do not include an explicit, comprehensive rationale. Without a rationale, the use, maintenance, and revision of standards is unnecessarily difficult. We introduce a new concept for standards, the rationalized standard. A rationalized standard combines: (a) an explicit goal defining a property desired for conformant systems, (b) guidance that, if followed correctly, should yield an entity with the property stated in the goal, and (c) the rationale showing the reasoning why there is assurance with reasonable confidence that a conformant entity will have the property defined by the goal. We illustrate the utility of an explicit rationale using an existing safety standard, ISO 26262.

Keywords: Standards, system safety, rigorous argument

1 Introduction

Safety standards such as ARP 4754 [1], ARP 4761 [2], IEC 61508 [4], and Mil Std 882E [9] have served the community well. They provide a repository of expert knowledge, foster consistency in the community, and document the expectations that arise when regulating agencies use standards conformance as a basis for approval. The role of a standard as a knowledge repository is especially important, because the knowledge usually originates from many experts and is subject to analysis and synthesis before becoming part of the standard. No single engineer or a small group is likely to have the composite background and experience of those who developed an officially accepted safety standard.

Despite their value, safety standards have been criticized in a variety of ways [3, 7, 8]. Standards have difficulty addressing the needs of individual systems or particular circumstances, and rarely define precisely what conformance will mean or how it will be assessed.

The most serious weakness with existing standards, however, is that, in almost all cases, conformance to a safety standard does not lead to assurance that the conforming

system has any specific property or properties other than those required to demonstrate conformance. A conforming system is viewed as generally suitable for its intended use, but use of a conforming system is based on an assumption that the system has one or more specific, implied but unstated properties.

At the heart of this problem is the fact that standards almost never include explicit documentation of the *rationale* for their contents. Developers produce evidence artifacts defined by the standard, and conformance experts examine these artifacts. With no explicit rationale for the evidence, both developers and conformance experts have to rely on their own knowledge of why the standard calls for certain forms of evidence and can do little more than check the form of the evidence produced. Frequently, conformance reduces to an almost meaningless checklist activity that ignores the technical *intent* of the standard.

In this paper, we introduce the *rationalized standard*. A rationalized standard combines: (a) an explicit *goal* defining a property desired for conformant systems, (b) *guidance* that, if followed correctly, should yield an entity with the property stated in the goal, and (c) the *rationale* showing the reasoning why there is assurance with reasonable confidence that a conformant entity will have the property defined by the goal.

The intent of a rationalized standard is *not* to simply improve the rigor of a standard; the intent is to change the relationship between a standard and the associated prerequisite knowledge in a fundamental way. Changing this relationship leads to a new and rigorous mechanism for using a standard.

The focus of this paper is safety standards although we note that the analysis is applicable to standards in general. This paper is organized as follows. In the next section we discuss the detailed circumstances of current standards, and in Section 3 we present the details of rationalized standards. In Sections 4 we illustrate the potential value of an explicit rationale by examining an existing safety standard, and in Section 5 we present our conclusions.

2 Current Standards

2.1 Development of Standards

An overview of the way in which existing standards are developed and applied is shown in Fig. 1. Most standards are developed by committees. A group of experts, the authors of the standard, convene based upon a perceived demand for a standard in the associated technical area. Each expert brings his or her own knowledge, both tacit and explicit, to the committee's deliberations.

The content of the evolving standard is the result of group meetings, discussions, drafts of the standard, white papers, etc. During these deliberations, individual elements of the standard are proposed, examined, refined, and either accepted or rejected resulting in a collective view of what the standard should contain. This view is then refined into a documented entity consisting of rules and guidelines. These rules and guidelines in turn define evidence that must be provided by developers using the standard to demonstrate conformance. The *intent* is that conformance to the framework will

assure particular qualities of engineering entities such as development processes, development artifacts, or complete systems. But the rationale for the content of the standard, though it existed, at least in the minds of the authoring committee members, is not available to either developers or conformance experts. In fact, the rationale was essentially discarded once development of the standard was complete and the committee disbanded.

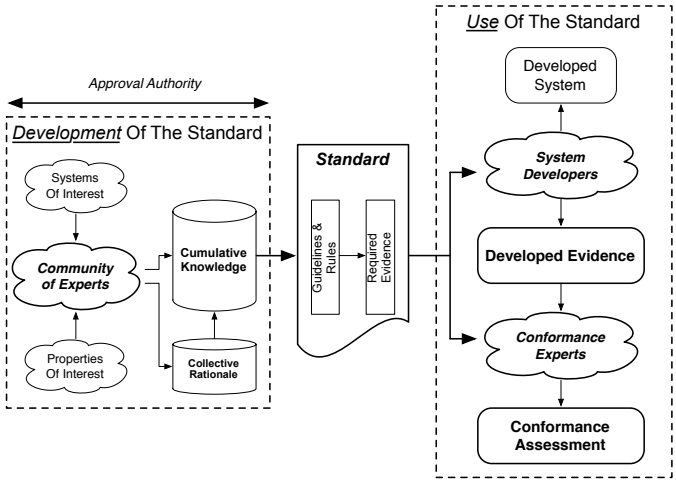

Fig. 1. Standards development and use as currently practiced.

2.2 Using Standards

Once developed, standards are put to use. No matter how careful the development process was, questions arise about interpretation, applicability, conformance, etc. To illustrate the difficulties that arise over time with current, static standards we examine a standard that was widely used for software assurance in avionics systems until recently, RTCA DO-178B [10]. Our use of this particular standard as an example is not intended to be critical of DO-178B in any way.

DO-178B was published in 1992 and replaced by DO-178C [11] only in 2012. During the twenty years that DO-178B was in effect, several documents were produced to supplement the standard including:

DO-248B. This guidance document is entitled "Final Annual Report for Clarification of DO-178B 'Software Considerations In Airborne Systems And Equipment Certification'" and was published in 2001 [12]. It was developed in response to hundreds of questions about DO-178B.

FAA Order 8110.49. This FAA order is entitled "Software Approval Guidelines". It provides a great deal of guidance on the software approval process, i.e., how compliance with DO-178B should be judged.

Certification Authorities Software Team (CAST) Position Papers. CAST papers are neither official policy nor guidance and are provided for educational and infor-

mational purposes. Nevertheless, they do play a role in supplementing the DO-178B standard. Twenty six CAST papers have been written to support DO-178B.

FAA Advisory Circular 20-148. This document is entitled "Reusable Software Components" and provides guidance for software reuse in the context of DO-178B.

Clearly the need for clarification in many different areas arose when the DO-178B standard was applied. We hypothesize that having an explicit rationale would have reduced the need for supplementary material substantially.

2.3 Maintenance of Standards

As is clear from the example in the previous section of the supplementary material that has been developed for DO-178B, those using a standard are likely to find defects, omissions or limitations. In addition, advances in technology will occur that could bring value to developers, but such advances are often prohibited because an existing standard does not address the new technology. Both of these circumstances motivate the need over time to modify the standard in some way.

Modification of a standard is closely related to developing the standard in the first place. The key difference between the development and subsequent modification of a standard is that the latter activity has one additional input, the standard itself.

Modifying a standard successfully requires a deep understanding of the standard and all of the technology that the standard references. Thus, as with development, modifications are: (a) often undertaken by a committee of experts, and (b) occur infrequently because of the difficulties and resource levels required.

Clearly, those undertaking a modification to a standard require access to the "why" of the content and the existing form of the standard in order to understand fully the ramifications of a modification. In other words, those undertaking a modification require access to the usually unavailable rationale for the original standard.

3 Rationalized Standards

3.1 The Concept

We conclude from the previous section that the explicit rationale for a standard as an integral part of the standard could provide great value. How then could the rationale be included in a standard? The rationale cannot merely be a new section added to an existing standard structure with no other change. The overall structure used for standards needs to be revised to both accommodate and take advantage of the introduction of the rationale.

In this section we introduce the *Rationalized Standard*. A Rationalized Standard emerges from recognition that the rationale for a standard is the *fundamental* content of the standard. The rationale is neither precursory nor supplementary; it argues, from first principles backed by sound evidence, why a set of proscriptive and prescriptive guidance will assure with reasonable confidence that an engineering entity holds a certain property that is stated explicitly.

The explicit rationale moves the rationale from the realm of *informality* and *oral tradition* into the realm of *rigor* and *written tradition*. Furthermore, it *models* the guidance of the standard as part of the rationale. The introduction of the rationale would restructure the associated standard and integrate revised versions of much of the existing material. Thus, the rationale would not lead to a major increase in the length of the standard.

Fig. 2 shows the three key components of a Rationalized Standard. At the top of the figure is the *Desired Property* (or properties) to be assured for a conformant system. At the bottom of the figure is the *Guidance* that defines conformance. In between is the *Reasoning* for why conformance engenders belief in the Desired Property.

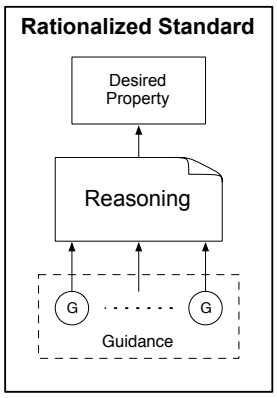

Fig. 2. The three key components of a rationalized standard.

The guidance specifies a set of items of evidence that must be obtained about the subject system by the system's developers. If the evidence is supplied and is sufficient, i.e., the subject system is determined to be conformant, then the reasoning leads to belief in the desired property. Ensuring that the reasoning is adequate to justify this belief is the responsibility of the authors of the standard.

The manner in which a rationalized standard would be used is shown in Fig. 3. Developers reference the reasoning in the standard in order to determine the suitability of the standard for their application and to map the evidence specifications in the guidance to their application. Conformance assessment requires judgment as to whether the evidence supplied meets the specification in the guidance.

3.2 Defining Reasoning

A convenient way to document reasoning is as a rigorous argument, and a hierarchical structure known as a *goal structure* is an effective way to represent an argument. A goal structure begins with a top-level goal that is decomposed into sub-goals such that belief in the top-level goal is justified by belief in the sub-goals using a documented strategy that links the sub-goals to the top-level goal. In the case of a standard, the top-level goal is the desired property. Each sub-goal in the goal structure is then decomposed into sub-goals that are further decomposed, and so on until leaf sub-goals are

reached. A leaf sub-goal is a goal for which belief can be justified directly by supplied evidence.

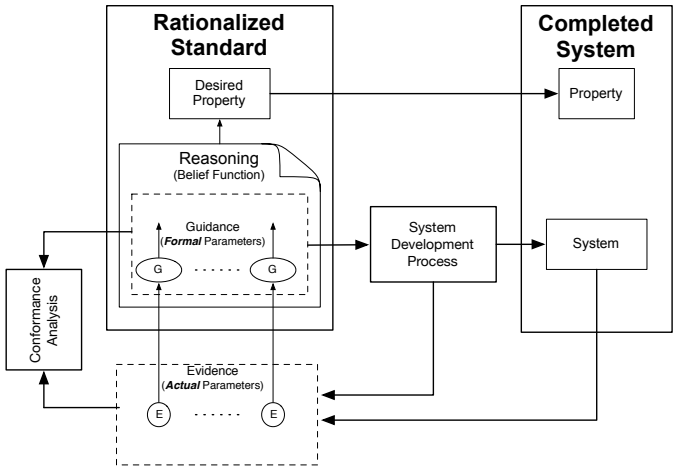

Fig. 3. Practical application of a rationalized standard.

Various notations have been developed for documenting arguments, and one notation that is in common use is the *Goal Structuring Notation* (GSN) [6]. We use GSN to document arguments in the remainder of this paper. The syntactic elements of GSN that we use in this paper are: (a) rectangle: a goal, (b) circle: an item of evidence such as results of a software test activity, and (c) rounded rectangle: an item of context – rigorous arguments are defined for a specific context such as the system's planned operating environment(s).

3.3 Defining Guidance

Recall that guidance is the specification of evidence that, if supplied and determined to be sufficient, justifies belief in the associated leaf goal. There are two forms that guidance can take in this case:

1. A statement of the leaf sub-goal and the required level of confidence that the evidence will justify belief in the sub-goal.
2. A statement of the explicit evidence that is required for the sub-goal.

As an example, suppose that, as part of the rationale for a safety standard, a leaf sub-goal is that the system software will ensure real-time task schedulability with an ultra-high level of assurance. This sub-goal derives from the overall safety goal of the system, and safety will be compromised without ultra-high assurance of schedulability.

In this case, stating the sub-goal in order to define the evidence requirement (guidance form 1 above) is insufficient, because the level of confidence required is so high. The evidence needed for this sub-goal will not be sufficient if, for example, developers

chose testing as an appropriate form of evidence. In this case, the rationale will specify specific forms of evidence. For example, the use of a static form of scheduling such as a time-triggered protocol together with a proof of the static schedule.

The structure shown in Fig. 2 is similar to an assurance case [13]. This similarity is not a coincidence. In essence what is required of a standard is an assurance case that will document the reason for belief that a conformant system has a desired property. The difference between a rationalized standard and a traditional assurance case is that the former has to be reusable whereas the latter is system specific.

4 Analysis Of A Safety Standard

In order to illustrate the concept of a rationalized standard, we examine a small piece of an existing safety standard, ISO 26262 [5]. Our use of this particular standard as an example is not intended to be critical of ISO 26262 in any way.

ISO 26262 was published in 2011 and is organized into ten parts. Our examples come from Part 6 entitled: "Product development at the software level". For simplicity, we limit our examples to Automotive Safety Integrity Level (ASIL) D, the highest integrity level.

Table 1. ISO 26262 Part 6 Section 5.4.7

Topics To Be Covered By Modelling & Coding Guidelines		ASIL D
1a	Enforcement of low complexity[a]	Highly Recommended
1b	Use of language subsets[b]	Highly Recommended
1c	Enforcement of strong typing[c]	Highly Recommended
1d	Use of defensive implementation techniques	Highly Recommended
1e	Use of established design principles	Highly Recommended
1f	Use of unambiguous graphical representation	Highly Recommended
1g	Use of style guides	Highly Recommended
1h	Use of naming conventions	Highly Recommended

Footnotes:

a An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.

b Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.
Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
Exclusion of language constructs which could result in unhandled run-time errors.

c The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

4.1 Example Element

Section 5.4.7 of ISO 26262 Part 6 has no specific title but is part of Section 5.4 entitled "Requirements and recommendations". Section 5.4.7 addresses, in part, software design and implementation correctness. The majority of the content of Section 5.4.7 is a

table listing eight techniques, the use of which the standard defines as "highly recommend" for ASIL D applications. The bulk of that table is reproduced here as Table 1 and the following footnotes.

4.2 Analysis of Example Element

To begin the analysis, we attempted to reconstruct the rationale for the example element. Although no rationale is included in ISO 26262, we examined the subject element and tried to identify: (a) the properties that conformance to the element should yield, (b) the evidence that should be produced, and (c) the reason for belief that the properties follow from conformance.

The intent of ISO 26262 Part 6, Section 5.4.7 is stated as:

> *"To support the correctness of the design and implementation, the design and coding guidelines for the modelling, or programming languages, shall address the topics listed in Table 1."*

We infer that the property intended is "the correctness of the design and implementation". The intended strategy is for the user to apply coding and design guidelines from Table 1, with the user providing suitable evidence. How this guidance helps fulfill correctness of design and implementation is not presented and is left to the reader's intuition. For example, line 1b in Table 1 ("Use of language subsets") gives examples of the type of issue that might be avoided by restricting the use of certain language features but defines neither a sub-goal for this guidance nor a complete set of properties that are desired from this guidance.

Presently, ISO 26262 is built around the philosophy of "organized containers of best practice". Table 1 is a container of techniques related to "correctness of design and implementation" by a common theme of software tools and techniques. But it is neither exhaustive nor definitive with respect to "correctness", "design", or "implementation", nor tools or techniques. Other related containers, organized by *themes*, are presented elsewhere in ISO 26262. Collectively, the standard implies (but does not state) that the thematic containers, when taken as a whole, will yield adequately safe software.

An example of the difficulties that arise with implicit arguments occurs with item 1b and footnote b – they should be *reversed*. The statement of "how" (use a language subset) is placed before the statement of "what" (avoid specific classes of fault). In practice there might be other techniques that could avoid the fault classes.

As a second example, consider that Line 1c in Table 1, "Enforcement of strong typing". This line merely indicates "how" one could reduce software faults, with no indication of "why" strong typing impacts the subject system. An implied community understanding of the utility of strong typing does not facilitate a comprehensive and uniform understanding of the role that it plays.

The standard does not define the evidence that is expected in order to conform to the guidelines. Evidence is crucial if rigorous (possible independent) conformance checking is to be undertaken. The standard needs to either define evidence that would be considered sufficient or define precise specifications of sufficient evidence.

The standard notes:

"Coding guidelines are usually different for different programming languages."

This observation is correct but does not preclude requiring, for a specific use of the standard, the development of one or more sub-goals for the associated guidance together with an appropriate language subset definition. Some languages are in common use and could be addressed specifically by the standard as an example.

Finally, we note that terms such as "complexity", "language subset" and "naming convention" have intuitive meanings, but intuition is not sufficient in dealing with challenging issues such as software correctness. ISO 26262 Part 1 is composed entirely of the definitions of terms but does not include definitions of these terms.

In summary, the important observations about this example are:

The properties of the software that are expected to follow from conformance to the guidance i.e., why the guidance should be observed, are not stated precisely.

The specific items of evidence that should be produced in order to justify conformance are neither defined nor specified.

The rationale for the guidance, i.e., why following the guidance implies a desired software property, is not stated. The standard is organized around collections of best practice and does not convey how these practices lead to a useful property.

The purpose of a rationalized standard is to provide a structure within which all of these issues are dealt with, so that a standard conveys why things add up, not just what adds up.

4.3 Rationalized Standard Fragment

Developing the rationale for the properties in the set listed in Table 1 is a tempting approach. But the existing ISO 26262 implied rationale is structurally weak for the reasons stated in Section 4.2 For example, important techniques could be missed, because they cut *across* themes.

A potential improvement would be to organize the rationale along a more compelling axis as shown in Fig. 4. In this organization, the top-level goal of the rationale is *fitness for use* in the target application, where fitness for use could reasonably be defined as: (a) the software meets stated requirements, and (b) the software avoids states that could lead to an identified system hazard. Combined, these indicate that the software does what is expected and prevents known hazards, and could be judged to be adequately safe. We note that avoiding identified hazards is an example of a cross-cutting theme related to "why" the system is safe. With this framing of the top-level goal, a decomposition into sub-goals is carried out based on best practices.

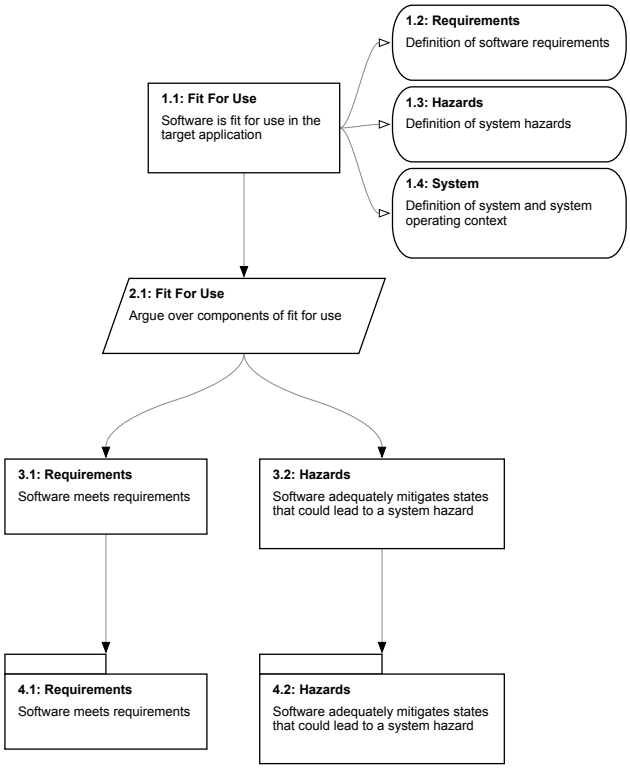

Fig. 4. Top-level rationale arguing fitness for use defined using the Goal Structuring Notation [6].

The lowest level of the goal structure for the refined rationale shown in Fig. 4 consists of two modules (4.1 and 4.2). A module is merely a means of encapsulating an argument fragment that is defined elsewhere. Rather than showing the content of these modules in GSN, for purposes of illustration, we provide an informal, text description of a possible goal structure for the "Requirements" module.

The goal "Software meets requirements" (the Requirements module) might be refined to two sub-goals: "Software meets functional requirements" and "Software meets non-functional requirements". An effective way to argue that a software entity meets functional requirements is to argue: (a) that the defined functionality is provided, and (b) the absence of faults in the software. Thus, the sub-goal "Software meets functional requirements" might be refined into a set of sub-goals; one for each element of functionality and one for each element in a taxonomy of fault classes. Each fault class would be addressed in the standard's guidance by suitable techniques of fault avoidance or fault elimination. This finally brings the goal structure down to a suitably low level of abstraction that specific techniques such as those in Table 1 can be introduced.

An example of this type of rationale is shown in Fig. 5. The black "dot" on the inference linking nodes 4.1 and 5.1 indicates repetition; a means of abstraction in the

rationale. The associated guidance should be applied to each functional element of the applicant system, as provided by the user.

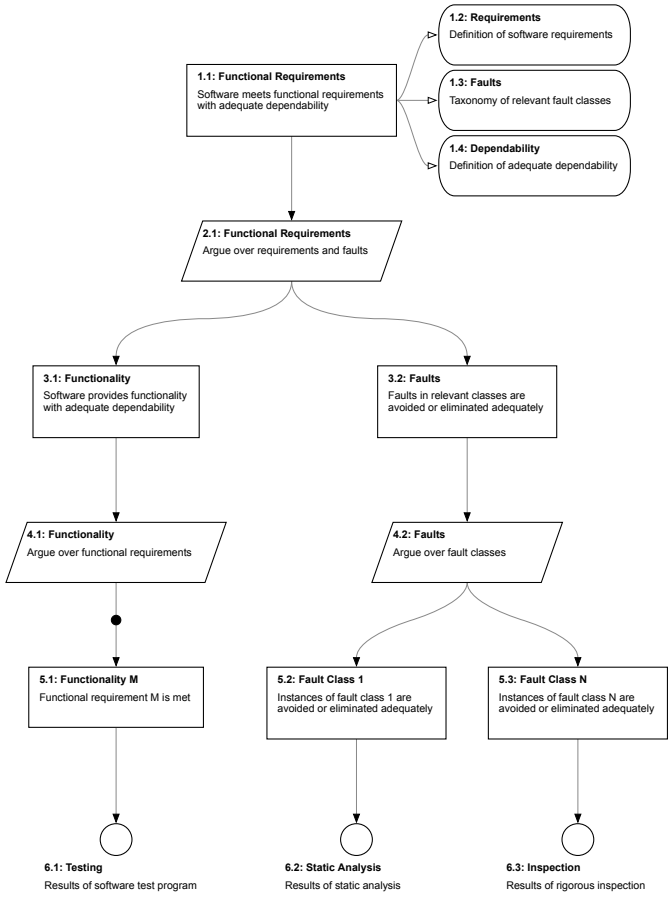

Fig. 5. Partial rationale for sub-goal "Software meets functional requirements"

5 Conclusion

Despite the value of existing safety standards, introducing an explicit rationale into a standard has the potential to improve the clarity, utility, and adaptability of standards considerably. These improvements emerge because the rationale for a standard becomes explicit information, retaining analyzable knowledge about the standard. The rationale enables incremental change as well as comparison and fitness with other rationales. Communities of practice can interact analytically with both the justification for the standard and with demonstration of conformance.

A rationale is intended to shed light on the "why" of a standard, material that is routinely discarded once a standard is published. The net result of explicit rationale,

therefore, should be a significant increase in the efficacy of the application of standards, their maintenance and enhancement, and the associated conformance assessment.

Introducing the rationale to a standard might seem likely to increase its length. This might happen, but the impression we have formed from analyzing ISO 26262 is that the rationale will replace large volumes of text rather than merely be an addition.

The availability of the rationale facilitates community discussion about the technical content of the standard and possible desirable enhancements. Valid and desirable enhancements are inevitable and would be integrated into periodic, controlled releases.

Finally, we note that the rationale is not meant to introduce prescriptive techniques that would be required for conformance. The evidence documented in the rationale would constrain developers only to the extent that would be necessary to ensure that the associated goals were met. Standards such as ISO 26262 already include recommendations for techniques and thus associated evidence. All that the rationale would do is to structure and justify such recommendations. Using alternative techniques would certainly be appropriate provided developers created a refined rationale that justified their technological choices.

Acknowledgment: This work supported in part by NASA Contract NNL13AA08C.

References

1. ARP4754: Guidelines for Development of Civil Aircraft and Systems, SAE International, 2010
2. ARP4761: Guidelines And Methods For Conducting The Safety Assessment Process On Civil Airborne Systems And Equipment, SAE International, 1996
3. N.E. Fenton, M. Neil, "A Strategy for Improving Safety Related Software Engineering Standards", IEEE Transactions on Software Engineering, Vol. 24, No. 11, November 1998
4. IEC 61508: Functional safety of electrical/electronic/ programmable electronic safety-related systems, International Electrotechnical Commission, 1998
5. ISO 26262: Road vehicles – Functional safety, Int. Organization for Standardization, 2011
6. T. Kelly and R. Weaver. "The goal structuring notation–a safety argument notation", In Proceedings DSN 2004 Workshop on Assurance Cases, Florence, Italy 2004
7. J. Knight, "Safety Standards – A New Approach", 22nd Safety-critical Systems Symposium, Brighton, UK, February 2014
8. C.Y. Laporte, R.V. O'Connor, L.H.G. Paucar, B. Gerancon, "An Innovative Approach in Developing Standard Professionals by Involving Software Engineering Students in Implementing and Improving International Standards", Standards Engineering: The Journal of The Society for Standards Professionals, Vol. 67, No. 2, April 2015
9. Mil-Std-882E, Department Of Defense Standard Practice System Safety, 2012
10. RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc. 1992
11. RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc. 2012
12. RTCA DO-248B, Final Annual Report for Clarification of DO-178B "Software Considerations In Airborne Systems And Equipment Certification", RTCA Inc. 2001
13. Software Engineering Institute, Assurance cases, Carnegie Mellon University, http://www.sei.cmu.edu/dependability/tools/assurancecase/