Adapting the Software Assurance Objectives Hierarchy to a Model-based Reuse Process

Deliverable 2: Mapping the Software Assurance Objectives Hierarchy (SOH) and Risk Informed Safety Case (RISC) Concepts to Applicable Standards

Pat Castle
NASA Ames Research Center
Moffett Field, CA 94035

Ewen Denney
SGT / NASA Ames Research Center
Moffett Field, CA 94035

Martin Feather
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Ganesh Pai
SGT / NASA Ames Research Center
Moffett Field, CA 94035

October 4, 2018

Abstract

We extend our earlier work on identifying and mapping BioSentinel assurance evidence to the Software Assurance Objectives Hierarchy (SOH). Specifically, first we discuss the relationship between the SOH and the Risk Informed Safety Case (RISC) concepts. Then, we develop fragments of a RISC for BioSentinel, as the mechanism by which to elaborate the rationale and the contextual details that show how the identified assurance evidence items relate to the objectives identified in the SOH. Then we discuss how the SOH and the RISC concepts relate to the NASA Software Assurance Standard NASA-STD-8739.8, which is largely the basis for assurance of the BioSentinel Flight Software (FSW).

Contents

1 Introduction **3**

2 Background **3**
2.1 Software Assurance Objectives Hierarchy . . . 3
2.2 Risk Informed Safety Case . . . 4
2.2.1 Overview . . . 4
2.2.2 Describing a RISC . . . 4
2.3 Mapping BioSentinel Assurance Evidence to the SOH . . . 5
2.3.1 BioSentinel Assurance Evidence . . . 5
2.3.2 Example Mapping . . . 9

3 Relating the SOH and RISC **9**
3.1 System Safety Framework . . . 9
3.2 Conceptual Mapping . . . 11

4 Developing a RISC for BioSentinel FSW Assurance **12**

5 Mapping the SOH to Applicable Standards **16**

6 Conclusion **18**

A Acronyms **20**

List of Figures

1 Top level of the draft SOH . . . 3
2 Goal Structuring Notation . . . 5
3 Software quality objectives applicable to BioSentinel . . . 9
4 System safety framework . . . 10
5 Software quality objectives hierarchy . . . 12
6 Software quality objectives hierarchy in GSN . . . 13
7 Refinement of software quality objectives . . . 14
8 Decomposition of software quality objectives . . . 15
9 Fragment of textual mapping . . . 17
10 Example SOH–software assurance standard mapping . . . 18

List of Tables

1 Assurance evidence from system engineering . . . 6
2 Assurance evidence from KDPs and reviews . . . 7
3 V&V-related assurance evidence . . . 7

1 Introduction

The overarching goal of this project is to characterize the nature and the extent of assurance afforded through the Software Assurance Objectives Hierarchy (SOH) [1] and Risk Informed Safety Case (RISC) [2], [3] concepts, whilst exploring their adaptation to the context of a model-based reuse process, exemplified by the BioSentinel project[1], in particular its FSW. In this report, we summarize the work done subsequent to our earlier report [4] on the project. In particular:

- In Section 2, we first give a brief background on the SOH and RISC concepts. Then we summarize BioSentinel FSW assurance evidence items, after which we give an example mapping from the identified evidence to one of the objectives of the SOH. This section provides the context for the work described in the rest of this report.
- In Section 3, we elaborate on a plausible relation between the SOH and RISC concepts, also giving details on the framework with which RISCs are developed.
- Thereafter, in Section 4, we give fragments of a RISC, as applied in the context of the BioSentinel FSW, to illustrate the relation described in Section 3.
- Section 5 discusses the relation between the SOH and the NASA Software Assurance Standard [5], which largely forms the basis for the assurance requirements levied on the BioSentinel FSW.
- Section 6 concludes the report describing the next steps in this project.

2 Background

2.1 Software Assurance Objectives Hierarchy

The Software Assurance Objectives Hierarchy (SOH) [1] is an *objectives-based* approach that is intended to facilitate engineering freedom and ingenuity in (NASA) programs and projects, with a view towards achieving flexibility, agility, and cost-effectiveness by focusing on the goals to be obtained rather than prescribing requirements on the processes by which those goals are achieved.

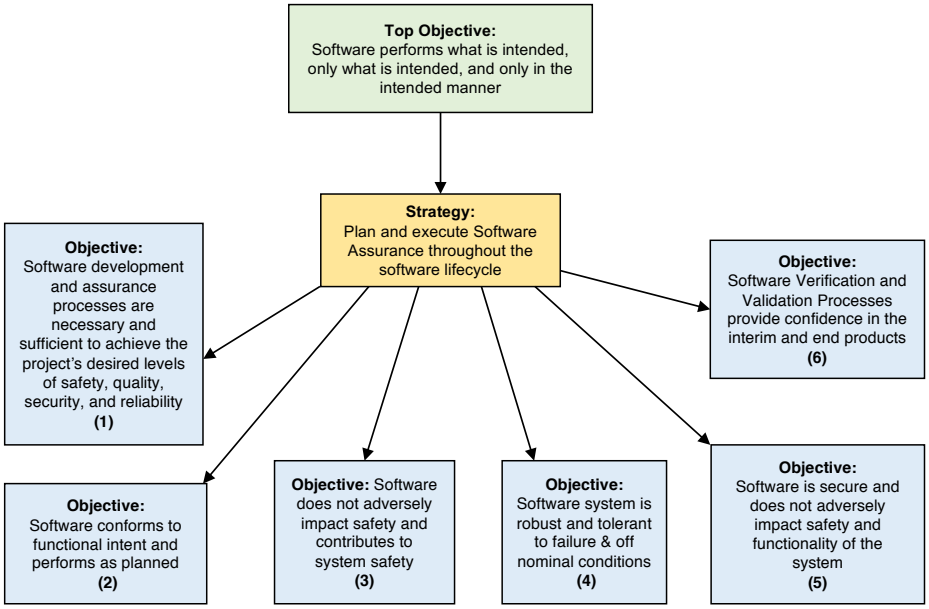

Figure 1. Top level of the draft Software Assurance Objectives Hierarchy (SOH), decomposing the overarching software assurance objective into supporting objectives addressing various concerns including conformance to functional intent, safety, reliability, etc.

The overall principle is to iteratively decompose the technical considerations that form the basis for the discipline (in this case, software assurance), into a hierarchy of objectives and strategies that build upon each other. Figure 1

[1] https://www.nasa.gov/centers/ames/engineering/projects/biosentinel.html

gives the top level of the SOH, showing how the overall assurance objective—software meeting its intent—has been decomposed into objectives concerning (1) process and planning, (2) conformance to functional intent, i.e., *quality*, (3) safety, (4) reliability and maintainability, (5) security, and (6) Verification and Validation (V&V) and IV&V.

Rather than mandating specific tools or methods, the objectives at lower hierarchy levels are concrete enough to allow an appropriate selection of the same, suitable to the engineering system to be built and its operating environment.

2.2 Risk Informed Safety Case

2.2.1 Overview

The concept of Risk Informed Safety Case (RISC) that we consider here is that which has been elaborated in detail in the NASA System Safety Handbook [2], [3]. In brief, a RISC is

> "*a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is or will be adequately safe for a given application in a given environment.*"

Note that this is, itself, an adoption of an earlier (but still in use) concept of safety case [6]. The core elements of a RISC given in this way are:

- *Safety claims*, which articulate statements about the safety of the system that are, or ought to be, true;
- *Evidence*, which refers to the artifacts produced during system development, verification, and operation, from which certain facts can be inferred (i.e., lower-level claims); and
- *Structured arguments*, which are a chain of reasoning or inference that capture the rationale linking the evidence to the safety claims made.

We develop RISCs guided by a *system safety framework* that aids the production and organization of the necessary safety artifacts, whilst guiding the relevant safety activities [2]. The framework is applied to span the system development lifecycle, and the core activities are iteratively undertaken such that there is an evolution of the RISC. This is reflected through the development of structured arguments that represent the (state of the) RISC, at each of the Key Decision Points (KDPs) in system development. Later in this report (Section 3), we will describe the system safety framework in more detail towards elaborating the relation between the RISC and SOH concepts.

2.2.2 Describing a RISC

When creating a RISC, we can elaborate it using textual descriptions, specialized graphical notations such as the GSN [7], or with a combination of the two. In this report, we will use GSN (Figure 2) to show fragments of a RISC that relates BioSentinel assurance evidence to the claims obtained from (the objectives of) the SOH. Note that GSN provides structuring mechanisms such as modules and hierarchy [8], along with supporting notational extension, though here we are only concerned with non-modular structures and their corresponding notation. Now, we give an overview of the basic (non-modular) GSN.

An argument structure in GSN (Figure 2) contains a top-level (root) *goal* stating a safety claim. We develop goals into sub-goals using *strategies*, and continue goal development until there are elementary claims that can be connected to the available evidence, i.e., *solutions*. The structure also specifies the *assumptions* made, the *justifications* if any, e.g., for the strategies used or the sub-claims created, as well as the *context* in which the claims, strategies, and solutions are valid. We link goals, strategies, and solutions using the *Is Supported By* link (→) while context, assumption, and justification elements require an *In Context Of* link (⇸). GSN provides a graphical annotation (◊) for goals and strategies to indicate that they are *to be developed*, i.e., they are incomplete.

Figure 2 gives a simple illustrative example: here, the top-level claim, "G1: Failures of the LiPo battery system are acceptably tolerated", which is made in the context of the failure modes and effects analysis (FMEA) of the LiPo battery system (context node C1) is decomposed by two strategies, S1 and S2, which provide complementary arguments—i.e., over the identified failure modes, and of redundancy, respectively. The latter relies on an assumption of independence in failures of the redundant batteries (assumption node A1), but has not been further developed. The use of the former has been justified (in justification node J1), and results in two sub-goals: G2 (concerning the elimination of short circuits in the battery system), and G3 (concerning the acceptable mitigation of thermal runaway), respectively. The latter remains to be developed, while the former is addressed by evidence node E1, i.e., short circuit analysis.

Note that GSN nodes are intended to be pointers to more detailed information, with the description of the node summarizing those details. Thus, we can give detailed definitions/content externally, and link those to an appropriate

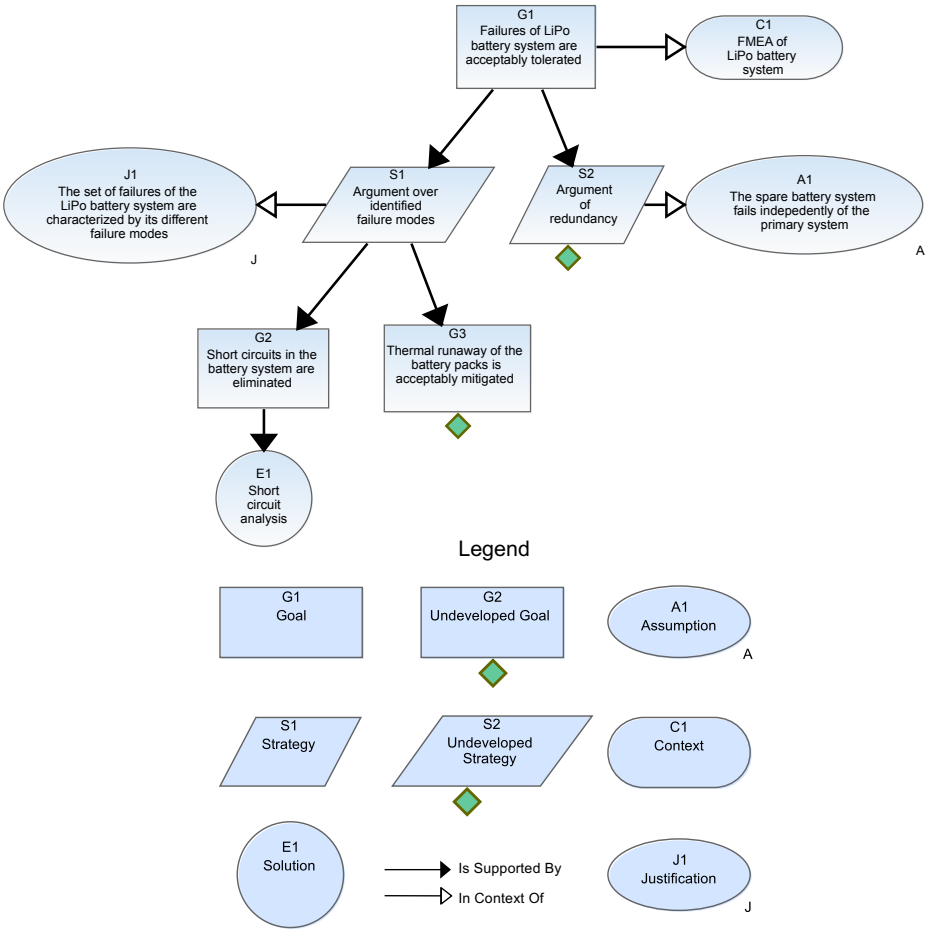

Figure 2. Basic (non-modular) GSN and a simple example to illustrate how GSN is used to specify an argument structure of a RISC.

node whose description could be, simply, an identifier. For example, the context node C1 of Figure 2 contains a simple clarifying description, although the content to which it would be linked could be the detailed FMEA report.

2.3 Mapping BioSentinel Assurance Evidence to the SOH

Here, we reproduce from the earlier report of this project [4], the content relevant for the current report: a preliminary mapping from BioSentinel assurance evidence to the objectives identified in the SOH. The intent is to give context to the work that we will report on here: how the assurance evidence from BioSentinel is linked to the SOH via (fragments of) a RISC.

2.3.1 BioSentinel Assurance Evidence

Tables 1–3 summarize (our understanding of) the assurance evidence produced by the BioSentinel project—i.e., the artifacts that document the processes being followed, and the concrete outputs of V&V processes—that are relevant for FSW assurance. Specifically, the evidence items listed in

- Table 1, pertain to assurance obtained from systems engineering, that process activities are in place to facilitate FSW development that conforms to the applicable assurance basis and standards.
- Table 2, concern the assurance obtained from systems engineering reviews conducted at different KDPs and milestones. Effectively, these reviews reflect assurance from a systems engineering standpoint that the FSW (releases) meets the requirements and, in turn, the stakeholders' needs.

- Table 3, elaborate the assurance afforded by V&V activities performed during the different FSW development stages. In effect, this constitutes *product-specific* evidence.

Additionally, *implicit* evidence is relied upon, as a consequence of software reuse, compliance of reused software with existing standards.

Table 1. FSW assurance evidence from BioSentinel systems engineering plans.

Plan	Description and Assurance Provided
Systems Engineering Management Plan (SEMP)	Describes the systems engineering process, guiding the overall technical engineering aspects and workflow, and provides the context for, and links to, supporting processes. Provides assurance that there is a common roadmap for system (and software) development that is structured, containing activities that are commensurate with the project/payload risk class.
Safety and Mission Assurance Plan (S&MAP)	Describes the tasks required for directing and controlling the design, development, review, and verification procedures and practices, so that BioSentinel meets its requirements for its intended lifetime. Provides assurance that a Software Quality Assurance (SQA) role has been designated, and tasked with responsibilities pertaining to creating, executing, and maintaining software assurance plans, processes, and products, as well as with oversight/audit responsibilities.
Software Assurance Plan (SAP)	Describes the activities planned to ensure conformance of the FSW to its requirements and the applicable assurance standards. Supplements the assurance provided by the software development plan that there is a roadmap and a framework to establish conformance of FSW to its requirements.
Fault Management Plan (FMP)	Describes the approach to, and scope of, BioSentinel fault management capabilities, elaborating the fault scenarios of interest, the extent to which fault monitoring occurs, and the roles/responses of the FSW in fault management. Strictly speaking, the FMP serves to provide *system-level assurance* that plausible faults that may occur during spacecraft operations (from pre-launch through decommissioning) have been anticipated, and that systems engineering activities (and artifacts) exist to manage/mitigate their effects. Nevertheless, we include this artifact amongst the evidence for software assurance, since the FMP characterizes the contribution of the FSW to system-level fault management, thereby describing a context where software performs potentially critical functions.
Software Development Plan (SDP)	Describes how BioSentinel onboard FSW is developed and verified, including specific assurance and V&V tasks to be undertaken. Provides assurance that there is a common roadmap for FSW development that is structured, containing activities that are commensurate with the software class. The SDP also includes guidance pertaining to software configuration management, development, integration, and release, together with the applicable workflows.
Software Release Plan (SRP)	Elaborates the schedule with which FSW is released to the project, along with a description of the purpose of each release and the functionality provided by the same. Provides assurance that *i*) the FSW development is following the SDP, and *ii*) the functionality delivered meets the applicable requirements, through traceability evidence that links the requirements to their verification (testing) results.

Table 2. FSW assurance evidence from BioSentinel systems engineering KDPs and milestone reviews.

KDP or Milestone	Description and Assurance Provided
Preliminary Technical and Safety Review (PTSR)	A system-level KDP review by the key stakeholders, of the core technical products, as specified in the SEMP. Applicable FSW-related artifacts include baselined SEMP, SDP, S&MAP, preliminary Interface Control Documents (ICDs), preliminary FSW subsystem (L4) requirements with verification methods, and an initial FSW architecture. The review provides assurance that FSW products conform to the assurance basis and standards. Reviewing facilitates early detection of errors, such as requirements specifications that do not meet the stakeholders' intent, or potential conflicts in the FSW architecture and the requirements.
Preliminary Design Review (PDR) and Critical Design Review (CDR)	System-level KDP reviews by the key stakeholders, of core technical products (developed after the PTSR KDP, as specified in the SEMP. Applicable FSW-related artifacts include FSW subsystem (L4) requirements with verification methods, preliminary FSW component (L5) requirements with verification methods, and the FSW architecture, all of which are baselined. Similar to the PTSR, the reviews provide assurance that the FSW products continue to conform to the assurance basis and standards, as they progress through development. As with the preceding KDP reviews, feedback is obtained on requirements errors, and the acceptability of the FSW architecture against the (L4) requirements.
Integration Readiness Review (IRR)	System-level milestone review of all BioSentinel subsystems, including the FSW, prior to final integration. Provides assurance that the FSW subsystem is ready for integration, by virtue of having completed and passed all unit test procedures, and all anomalies recorded in JIRA, relevant to the release being integrated.
Flight Readiness Review (FRR)	Project-level milestone review of the overall system, adjudging whether the system artifacts, including those pertaining to the FSW have been developed in conformance to the applicable assurance basis. The main assurance provided pertains to the completion of all verification items, internal consistency of all requirements, traceability to functions and items that fulfill the requirements, resolution and closure of all non-conforming item reports, and completion of all software (as well as hardware) end items.
SQA Audits	Strictly speaking, these are not conducted at KDPs or at milestones; rather they are conducted as and when deemed necessary by the SQA role defined within the SEMP. Provides assurance that the SDP is appropriate for the identified software class, and additionally that the FSW products, documents and processes meet the requirements identified by the applicable assurance basis and standards.

Table 3. Assurance evidence from V&V activities during FSW development stages.

Name	Description and Assurance Provided
Architecture verification results	The algorithmic apps of the FSW undergo testing via the Workstation Simulator (WSIM), and limited formal verification through the use of Simulink Design Verifier (SDV). The corresponding results are V&V-generated evidence items that provide assurance that the algorithmic components in the FSW meet their allocated requirements.
Static analysis results	Static analysis using the compiler and third-party tools, such as Cppcheck, integrated into Bamboo provides assurance of freedom from certain kinds of runtime errors.

Table 3. Assurance evidence from V&V activities during FSW development stages (Continued).

Name	Description and Assurance Provided
Review results	Reviewing forms one of the core V&V activities conducted on the FSW requirements, architecture and design, and the code. The latter, in particular, undergoes both formal and informal review. The results of reviewing are reflected either as requirements that are baselined and eventually closed (i.e., verified), or as new issues that are recorded and tracked to closure in JIRA. The assurance concerns addressed include requirements validity, traceability correctness (i.e., that the right collection of artifacts is linked), requirements consistency, acceptability of the architecture against the requirements, conformance of the code to coding standards, reduction of errors in code, and the acceptability of the FSW releases in relation to stakeholder needs.
Traceability records	Requirements and any issues discovered during review (or in other V&V activities) are recorded and tracked to closure in JIRA. As such JIRA queries help generate issue closure reports, and ensure that FSW requirements are traceable not only to their parent/child requirements in the requirements hierarchy, but also to their verification procedures. The former provides assurance that the errors discovered have been fixed, while the latter provides assurance on the completeness of requirements traceability, allocation, and verification. In addition to JIRA, the Bamboo continuous integration and build environment provides the record of traceability from requirements to their unit test results, while the results of functional/system-level testing are recorded in Confluence, together with links to the FSW requirements that they verify.
Unit testing results	Code corresponding to L5 FSW requirements, i.e., the allocated FSW components, undergo unit testing. The record indicating the traceability from requirements to unit test/testing results is generated via Bamboo. Unit tests provide assurance that the individual components meet their associated requirements.
Functional testing results	FSW also undergoes functional testing, which includes tests that exercise the function being provided, as well as system-level scenarios that test the overall system, along with specific FSW functions. Functional tests verify some L5 FSW component requirements and all L4 FSW subsystem requirements. The record of test results and their traces to the requirements being verified is maintained and managed on Confluence. As such, these evidence items provide assurance that the executable FSW meets the relevant system and subsystem-level requirements.
JIRA reports	JIRA is also used to record issues discovered not only during review of the FSW development artifacts, but also during the other V&V activities conducted. Consequently, tracking all JIRA issues to closure, and the corresponding generated reports provide assurance of concerns such as requirements being satisfactorily verified, trends on reduction of errors in code, code quality improvement, etc., in addition to traceability between requirements, and between requirements and their verifications.
Thermal Vacuum and Thermal Vacuum Power Management testing results	FSW releases undergo Thermal Vacuum (TVAC) and Thermal Vacuum Power Management (TVPM) testing, the results of which are recorded and maintained on Confluence. Those results constitute evidence that provides assurance that the FSW will function as required in an environment representative of the actual mission operating environment.

2.3.2 Example Mapping

Figure 3 exemplifies how the various identified BioSentinel assurance evidence items have been mapped to the SOH of the *software quality objective* (see Figure 1). For other objectives of the SOH and how BioSentinel assurance evidence is related to those objectives, we refer to the earlier report of this project [4].

The purpose of this mapping is threefold:

i) from a *top-down* perspective, to show which objectives can be claimed to have been addressed by the assurance evidence gathered from BioSentinel assurance activities;

ii) to give a qualitative gauge of the extent to which the objectives have been met by the evidence shown; and

iii) to give insight into the relevance of the assurance evidence gathered to the objectives identified.

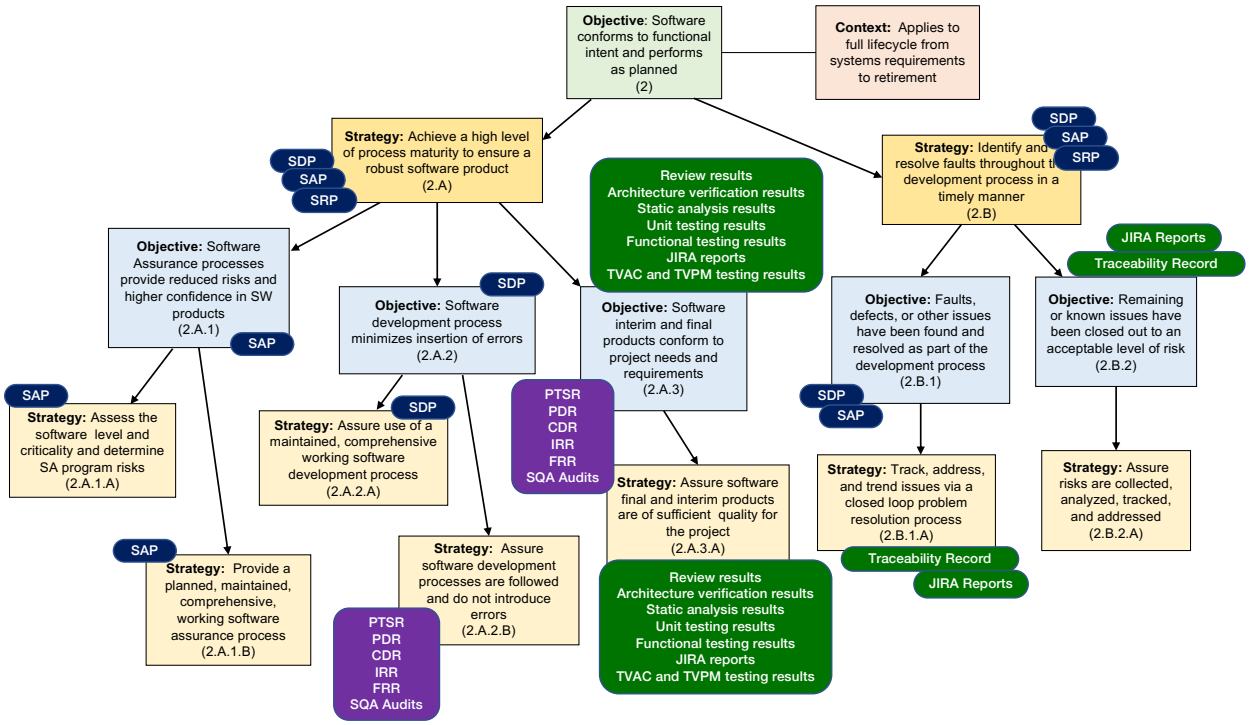

Figure 3. Draft SOH at a lower level, as developed for the software quality objective (2), with each sub-objective and strategy annotated with the BioSentinel FSW evidence items.

From Figure 3, we can infer that some objectives are related to the mapped evidence. That is, the objectives (appear to) have been addressed by the evidence items shown. However, the rationale why the evidence entails the claims is not explicitly apparent. The main focus in this report is to elaborate on this mapping between the relevant V&V artifacts and (the objectives of) the SOH, through a *skeleton* of an assurance argument. The argument captures the rationale why the evidence provided supports the identified objectives. An additional focus is to map the SOH to the current, applicable assurance standards.

3 Relating the SOH and RISC

As mentioned earlier (Section 2.2), we develop a RISC within a system safety framework. First we give an overview of this framework, after which we relate the SOH and the RISC concepts.

3.1 System Safety Framework

Figure 4, reproduced from the NASA System Safety Handbook [2], gives a pictorial overview of the system safety framework, whose core activities comprise development of safety objectives, system safety activities, and the development and evaluation of RISCs.

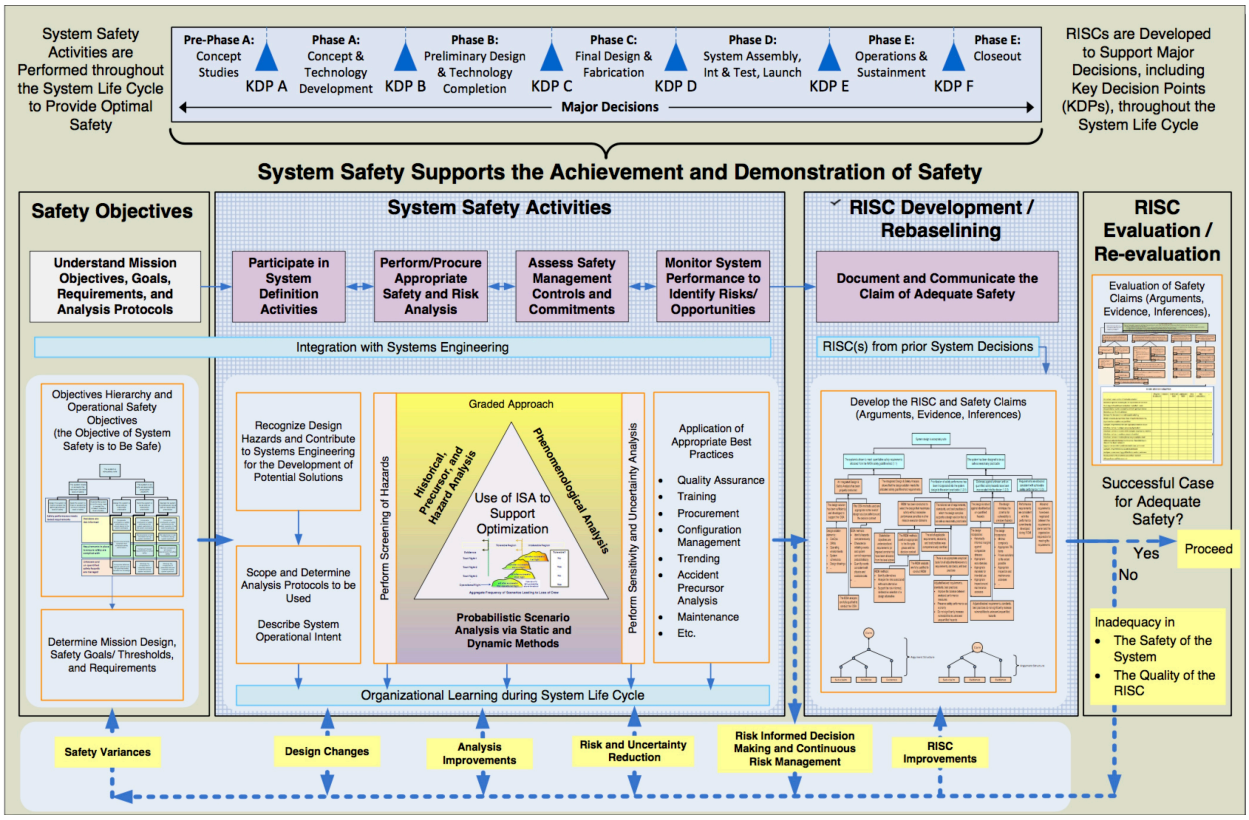

Figure 4. System safety framework, reproduced from [2], to develop RISCs.

Development of Safety Objectives: This activity produces a collection of related safety objectives, which can be expressed in the form of an objectives hierarchy. The hierarchy is produced by decomposition of the highest-level objectives into their *fundamental components*, e.g., decomposing a system safety objective into design safety, and operational safety. Lower level objectives in this hierarchy are then associated with measurable performance parameters, e.g., the probability of an unsafe malfunction. The safety objectives are specified such that the level of safety to be achieved goes beyond the minimum tolerable level. That is, the aim is, in fact, to maximize the achieved level of safety, without disproportionately deteriorating other system performance parameters.

System Safety Activities: This refers to a collection of activities to: *a*) develop and analyze hazardous scenarios—so-called *integrated safety analysis* (ISA); *b*) develop safety requirements; *c*) support the system design, i.e., by considering design decisions from a safety perspective; *d*) analyze alternatives; *e*) demonstrate that safety requirements are satisfied, both from the standpoint of the external requirements levied on the system (e.g., from regulations), and the lower-level requirements derived from preceding activities; *f*) monitor safety performance; and *g*) interface with other systems engineering processes that indirectly impact safety, e.g., quality management, organizational processes, etc. Of these, the *analysis of alternatives* is the core activity that makes the overall approach *risk-informed*. Specifically, the analysis considers the impact (on cost, schedule, and technical performance) of introducing safety risk reduction mechanisms with respect to the rest of the system performance. In other words, design decisions are made, in part, by considering the cost benefit trade-off between system safety and other orthogonal system properties, e.g., functional performance, reliability, etc.[2]

Among the results of analyzing alternatives in this way is a refinement of the safety objectives identified in the earlier stage. This induces a feedback loop in the process so that the lower-level objectives and performance parameters ultimately selected are the result of an iterative application of the system safety activities.

[2]The cost-benefit analysis lends to the notion of a system that is *as safe as is reasonably practicable* (ASARP), which is taken as a fundamental principle of adequate system safety [2].

Development and Evaluation of a RISC: Developing a RISC effectively amounts to the development of a structured argument that presents the chain of reasoning linking a hierarchy of safety claims to the evidence required to assert that the claims made can be accepted. In fact, the result of this activity from the application of the overall framework are a series of RISCs, for each KDP in the system development lifecycle (shown at the top of Figure 4). The structure of such arguments (as shown in the figure) is similar to the structure of the objectives hierarchy produced from the very first activity in the system safety framework. In fact, the hierarchy of claims can be developed to correspond exactly to the safety objectives hierarchy.[3]

Although the framework does not specify a particular notation or format for the argument structure embodying a RISCs, it borrows the concepts and structuring mechanisms from those of the GSN [7]. As mentioned earlier, in this report, we will use the GSN to develop the RISCs fragments for BioSentinel FSW assurance.

RISC evaluation is conducted in the system safety framework as a structured review by the appropriate decision makers, by considering the technical basis and evidence underlying a safety claim, its validity and its adequacy, supported by checklists that address various evaluation criteria. We will not further consider the methodological details of developing and evaluating RISCs in this report, which have been addressed comprehensively in [2], and [3]. We also refer to our prior work in developing aviation safety cases [9], [10], [11], [12], which provides additional details on the development of structured arguments.

3.2 Conceptual Mapping

We now describe plausible relations between the SOH and the RISC.

First, from Figure 4, we observe that the initial work product of applying the system safety framework is a safety objectives hierarchy. This is largely similar in intent and form (i.e., it specifies high-level objectives reflecting that which needs to be achieved by the system from a safety standpoint, without emphasizing how the objectives are to be achieved) to the SOH, although the focus is on system-level safety concerns.

It is conceivable that upon refining the system safety objectives hierarchy, there are objectives that apply to its software components. In other words, by allocating the objectives to the system architecture—in much the same way as requirements can be allocated to the system and its components—some of the objectives may apply to the software elements of the system. Those objectives, in turn, may be related to, (or even may be the same as) the objectives identified within the SOH [1] that apply to software quality, software safety, and software reliability and maintainability, and which have themselves been allocated to the software component.

Here, our rationale is that each class of software properties can have an impact on system safety. Therefore, it appears reasonable to hypothesize that there will be objectives in common between the SOH and a refinement of the system safety objectives hierarchy as allocated to software, so that in effect the former contributes to the latter. To test this hypothesis, we can apply the system safety objectives hierarchy to the BioSentinel project, allocating and refining them to derive the lower-level objectives and determine whether any apply to the BioSentinel FSW. At the same time, we can also apply the SOH, refining the objectives as applicable to BioSentinel FSW and compare the results. We do not undertake this effort in this project.

Another possible mapping is determined by applying the system safety framework to the BioSentinel FSW. That is, rather than deriving a system safety objectives hierarchy, we replace it with the SOH as the starting point, following which we apply the (system) safety activities to the BioSentinel FSW. Thus, one of the outcomes is an iterative refinement of the SOH into specific lower-level objectives relevant for BioSentinel. Applying system safety activities to software involves conducting a software-focused hazard analysis, based on which we can develop scenarios that help to determine when and how the FSW affects system safety. That, in turn, will help to determine whether or not the current FSW design and its implementation are sufficient for managing the identified software induced hazards, or whether additional risk reduction measures are needed. An additional task is (re)analyzing software-level design decisions from a safety perspective and, if warranted, modifying the software architecture (and implementation) to meet any new or derived safety requirements, beyond those levied as a result of the safety analysis that had been conducted at the system level.[4]

In this report, we do not apply the system safety framework to the BioSentinel FSW mainly because it has been determined to be non-safety critical, Class C, mission support software [4], therefore the application of additional

[3] Indeed, as per [2], a safety objective specifies that which is *intended to be achieved*, whereas a safety claim specifies that which *can be claimed to have been achieved*. Additional concepts include a notion of (safety) *threshold* and *goal*, which specify the upper and lower bound, respectively, of the safety target deemed to be the minimum tolerable.

[4] i.e., safety analysis applied to the BioSentinel Freeflyer satellite system, and the Space Launch System (SLS) in which it is a secondary payload.

safety analysis is not warranted. However, we *do* use the broad approach to develop (fragments of) the assurance argument, i.e., the RISC, to relate the assurance evidence to the objectives in the SOH. Thus, the RISC can be viewed as the means which we explicitly capture the rationale why BioSentinel assurance evidence items (Tables 1 – 3) can be used to conclude that the objectives in the SOH have been met (for example, as shown by the mapping in Figure 3).

Here, we note that—from a practical standpoint—we have found it beneficial to expand the scope of a RISC to include a concept of *safety architecture* [13], in addition to structured arguments. Effectively, a safety architecture describes a composition of safety scenarios showing the safety mitigations being used, where the scenarios are described using *Bow Tie Diagrams* (BTDs) [14]. Recall that safety scenarios are amongst the outcomes of the system safety framework, in particular the ISA, as described earlier in Section 3.1. We do not develop scenarios for the BioSentinel FSW in this report.

4 Developing a RISC for BioSentinel FSW Assurance

We now present fragments of the assurance argument, i.e., the RISC, relating BioSentinel assurance evidence to the SOH.

Recalling the observations in our earlier report [4], the SOH in its current form (intentionally) states its objectives in a generic form. To apply it for a particular software system, the objectives are to be specialized and refined referring to the software for which assurance is sought.

As mentioned in Section 2.2, we will use the GSN to describe the argument that embodies the RISC to capture the rationale why BioSentinel assurance evidence items support the objectives stated in the SOH. In particular, we apply a combination of *instantiation*—i.e., translating the SOH into GSN—and *specialization*—i.e., a refinement where we reword the generic objectives into application-specific *claims*, also modifying the SOH strategies into appropriate assurance and inference strategies. The link to evidence is made through decomposition of the claims, whilst capturing assumptions, justifications, and context.

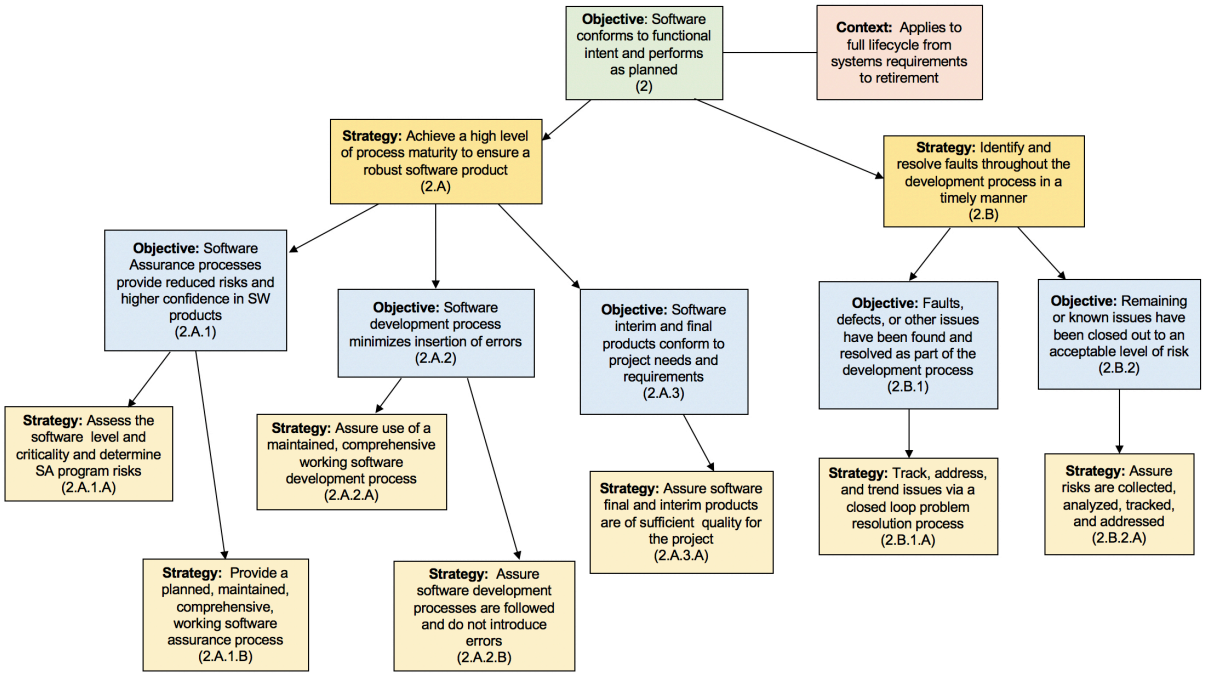

Figure 5. Software quality objectives (sub-) hierarchy.

Specialization of the SOH Figure 5 shows the software quality objectives (sub-) hierarchy, while Figure 6 shows the same hierarchy described using GSN, with the node identifiers of the latter retaining the identifiers of the objectives and strategies of the former. The main difference between the two is the use of the appropriate GSN, together with an indication of incompleteness (i.e., the ◊ decoration on leaf strategy nodes) in Figure 6.

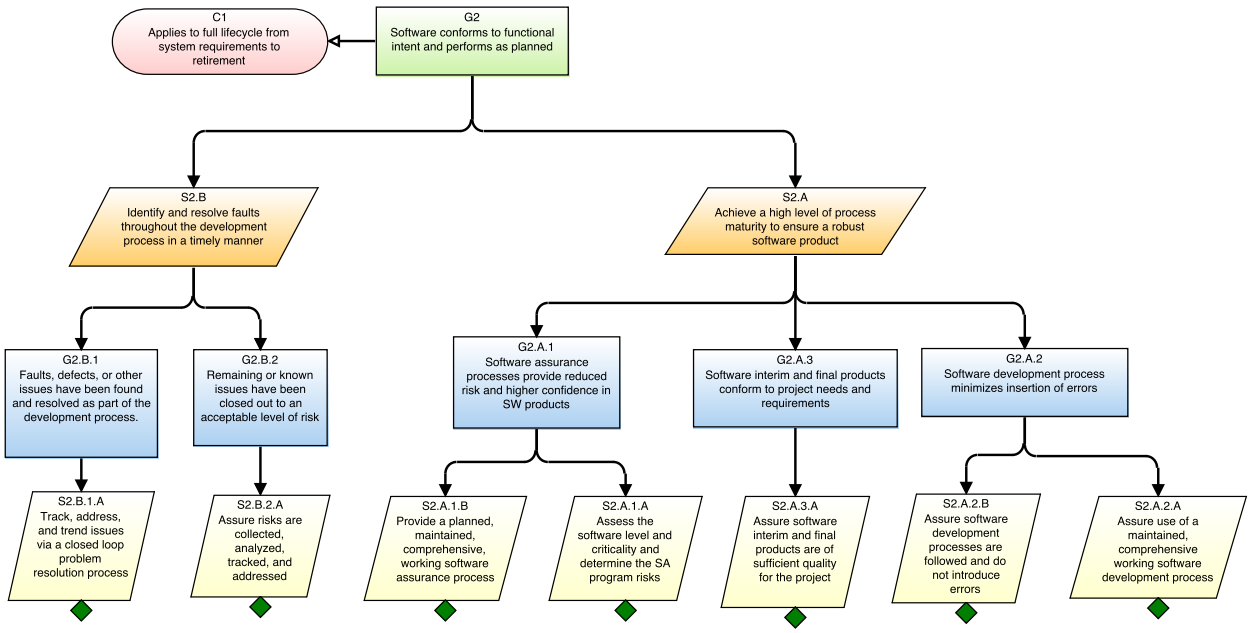

Figure 6. Software quality objectives (sub-) hierarchy described using GSN. Note that the nodes have been colored to retain the same color scheme as the hierarchy of Figure 5 to facilitate a rapid visual comparison.

Refinement Figure 7 shows a refinement of the argument structure of Figure 6. In the refined argument structure, we use node colors to indicate node type (i.e., objective, strategy, context) in the original hierarchy, with the identifier and its suffix showing the corresponding source in the SOH:

- Node identifiers whose suffix includes the letter 'R' are nodes of the hierarchy that have undergone a specialization of the description appropriate for capturing a claim, strategy, or context. For example, the root goal G2 of Figure 6 states the objective "*Software conforms to functional intent and performs as planned*"; its refinement, the root goal G2R in Figure 7, specializes this objective into a claim relevant for BioSentinel, i.e., "*BioSentinel FSW conforms to functional intent and performs as planned*".
- Likewise, the strategy S2.B in Figure 6 (*Identify and resolve faults* . . .) has been refined into the strategy S2.BR in Figure 7) (*Appeal to timely identification and resolution of faults throughout the development process*.
- In Figure 7, the nodes whose identifiers are, respectively, G2.B.1R1 and G2.B.1R2, are both goal nodes and refinements of the objective G2.B.1 (see Figure 5, and its equivalent GSN argument Figure 6).
- Additional nodes that refine the rationale being captured have no color. For instance, the context node C1 in Figure 7, is additional context for the claim in goal node G2R, i.e., the determination of conformance to functional intent requires the (BioSentinel FSW) functional specification—which documents the intent—as context. Similarly, the nodes G1, G2, and G3, are sub-claims of the claim in the goal node G2.B.1R1. Since they are linked to GSN solution nodes, they are also *evidence assertions*, i.e., they state that which can be inferred from the relevant evidence items. For instance, the solution node E1 refers to a traceability record produced from JIRA queries, from which it can be established that: *i*) there is a listing of all recorded FSW issues, including faults and defects, *ii*) all the issues listed have been resolved and closed.

Based on this, in part, we can conclude that the BioSentinel FSW traceability record provides evidential support for the claim in goal node G2.B.1R1, and why that evidence item maps to the corresponding branch of the SOH (as shown in Figure 3). Similar rationale is captured in the remainder of argument structure shown in Figure 7. Note that the evidence to which the solution node E1 refers is software-level *product* evidence; the evidence in solution nodes E2–E4 refer to artifacts and infrastructure that constitute or characterize the development and planning processes.

Refinement and Decomposition The argument fragment of Figure 7 adopts some of the strategies provided in the SOH to link the root claim (shown in the goal node G2R) to the supporting assurance evidence. An alternative refinement is to consider the root claim in the context of the lifecycle artifacts (context node C1R).

Since lifecycle artifacts represent, in effect, the evolution of the final executable FSW starting from the require-

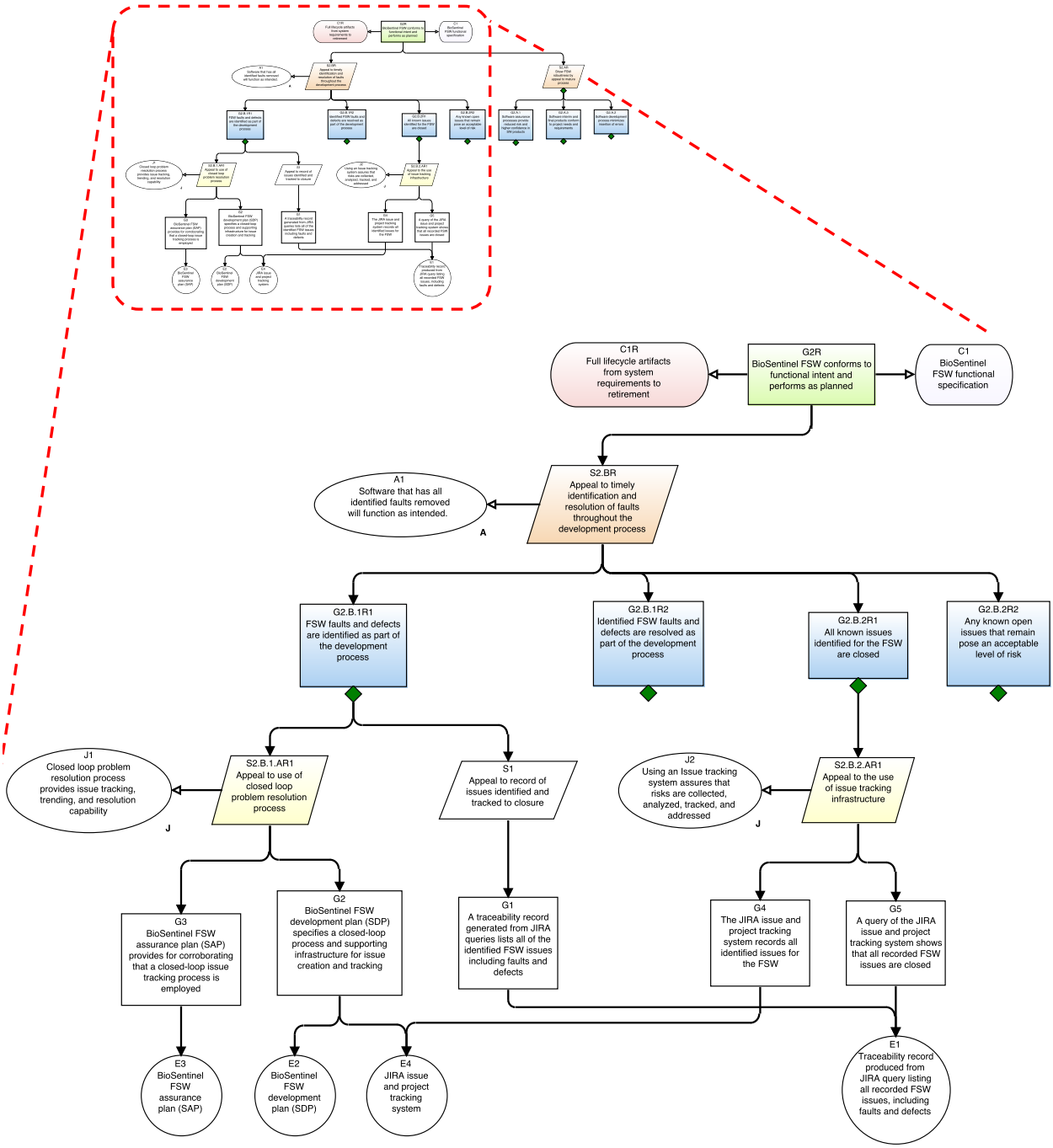

Figure 7. Refinement of the software quality objectives (sub-) hierarchy (shown at the top), with one branch (highlighted by the dashed box) shown expanded.

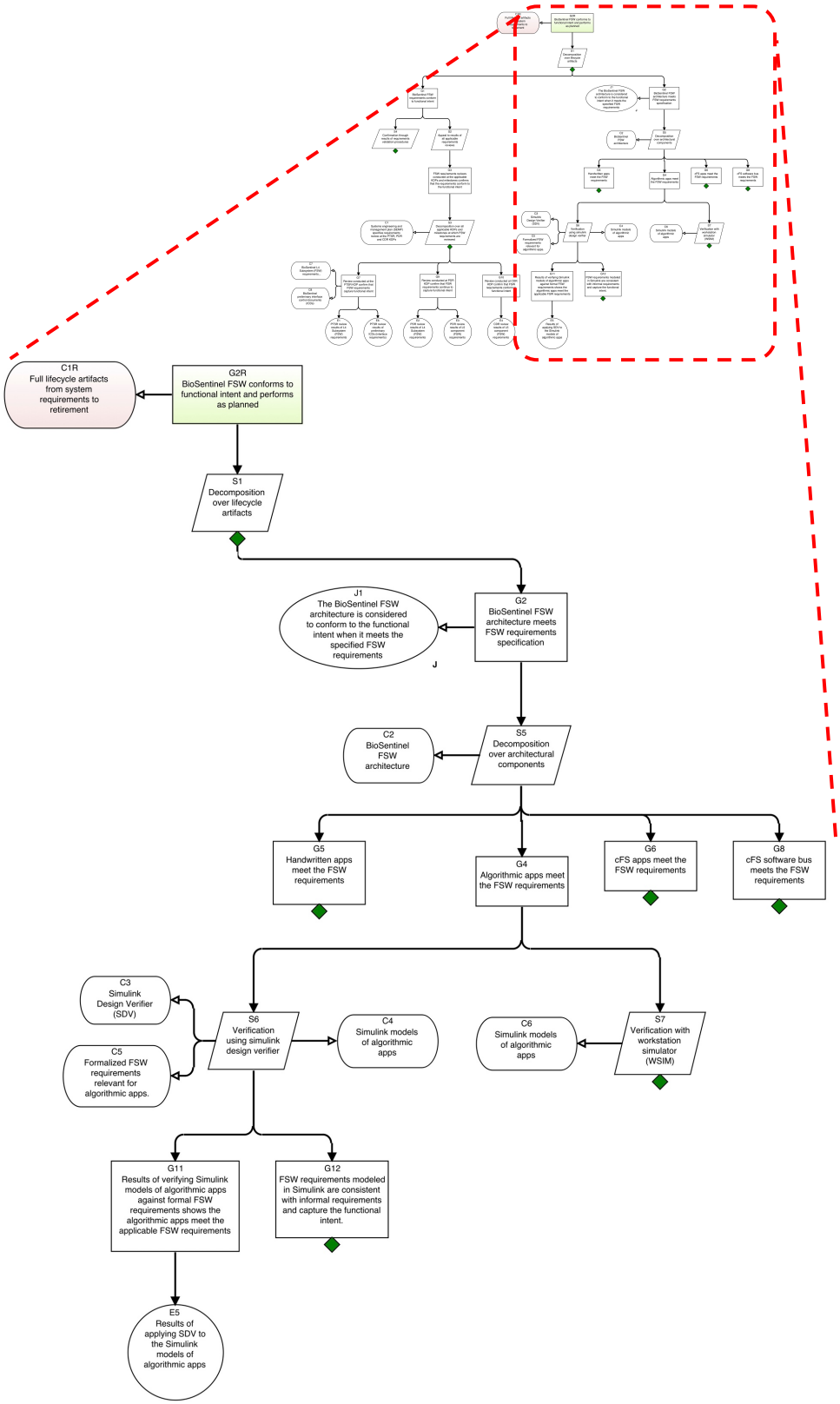

Figure 8. (Fragment of) an alternative argument for why BioSentinel assurance evidence items support the software quality objective (shown at the top), produced by decomposition and refinement, with the right-side branch (highlighted by the dashed box) shown expanded.

ments, it appears reasonable to consider that conformance to functional intent ought to also hold for each lifecycle artifact, with "conformance to functional intent" being defined in a suitable way for the artifact under consideration. That is, by decomposing the root claim over the lifecycle artifacts and showing that each artifact considered in turn itself appropriately conforms to functional intent, an alternative rationale can be provided to reinforce why BioSentinel FSW assurance evidence supports the SOH.

Figure 8 shows a fragment of this alternative argument. The format used for node identifiers and the node color scheme is as earlier, with claims, strategies and other elements of the argument not previously appearing in the SOH shown with no color. As shown, the argument uses decomposition iteratively, first over lifecycle artifacts (strategy node S1), and then over the FSW architecture (strategy node S5). The root claim of conformance to functional intent is accordingly refined into sub-claims on the FSW architecture (goal node G2), and the components of the architecture (goal nodes G4, G5, G6, and G8). The assurance evidence then used to substantiate those claims comprises, in part, verification evidence, e.g., formal verification applied to Simulink models (goal G11 and solution E5).

Overall RISC We have mainly shown fragments of the assurance arguments comprising the RISC, to give a flavor of how the arguments are constructed and the broad approaches (specialization, decomposition, and refinement) to link the claims to the assurance evidence. We do not develop the complete RISC here, in part, since the capture of assurance rationale at the required level of detail requires greater effort and in-depth analysis of the BioSentinel FSW than is feasible within the scope of this project.

5 Mapping the SOH to Applicable Standards

The main applicable standard for this work is the NASA Software Assurance Standard, NASA-STD-8739.8 [5], though other relevant assurance bases for BioSentinel have been elaborated in [4].

For BioSentinel FSW, the NASA Software Engineering Requirements (NSER) [15], and the NASA Software Assurance Standard (NSAS) [5], together establish the assurance basis in terms of

- the *software class*: a usage/application-based classification of the software, and
- the level of assurance effort required, and its prioritization, based on the software class.

Moreover, the NSER prescribes process activities that contribute to the provision of assurance—e.g., requirements on software V&V activities, such as verification planning and verification result tracking—while the NASA Software Assurance Standard (NSAS) defines the assurance activities and the tasks required to meet the applicable assurance objectives. Furthermore, the NASA software safety standard [16] establishes software contribution to system safety through a determination of software safety criticality, imposing additional safety-related software assurance requirements. As such, the BioSentinel onboard FSW has been determined to be *non safety-critical, Class C, mission support software*, i.e., per the current NSAS (Table A3, Appendix A), Class C software requires a *medium* level of assurance effort and *medium prioritization*.

The BioSentinel project has tailored the requirements originating from both the NSER and NSAS to be commensurate with the payload risk class and software class, as identified above. There are additional center-specific requirements—in this case, from the Ames Research Center (ARC)—which specialize and tailor the above Agency-wide requirements.

As stated earlier, the objectives-based approach underpinning the SOH contrasts with the prescriptive nature of the more traditional NSAS. Both are intended to direct assurance to provide confidence that the software resulting from the development process will serve its intended purpose. Thus, it should be possible to trace how satisfying the requirements of the NSAS will provide the evidence contributing towards the objectives of the SOH. Likewise, for traditional development it should be possible to trace how the evidence needs of the SOH call for the kinds of activities prescribed by the NSAS. In this project we have investigated this by creating a mapping (i.e., bi-directional trace) between the leaf strategy nodes of the SOH and the requirements of the latest *draft* of the NSAS. Note that the BioSentinel mission complies with the current active NSAS, and has not considered the new draft.

The NSAS-SOH mapping was provided earlier by this project in the form of a spreadsheet, and addressed all the SOH leaf strategies (except for those pertaining to cybersecurity) and all the draft NSAS requirements that are the responsibility of Software Assurance on a project (again except those pertaining to cybersecurity—beyond our expertise to address).

As an example, Figure 9 is a fragment of the textual listing of the portion of this mapping for the Objective (2) of the SOH—the branch of the SOH shown earlier in Figure 3 and Figure 5. The lines in **blue** are the textual contents

of SOH *objective* nodes, the lines in **red** are the textual contents of the SOH *strategy* nodes, and the lines in **black**, grouped beneath the lowest-level SOH *strategy* nodes, are the draft NSAS requirements (those levied on non-safety critical, Class C mission support software) that we assessed as contributing evidence towards their parent SOH strategy.

The "SAS-###" numbers are those in the current draft NSAS, and the associated text is, for reasons of conciseness here, a much-abbreviated form of the requirement's actual text—for the full definitions of the requirements, see the draft NSAS itself.

Software performs what is intended, only what is intended, and only in the intended manner
- **Plan and execute Software Assurance throughout the software lifecycle**
  - **2: Software conforms to functional intent and performs as planned**
    - **2.A: Achieve a high level of process maturity to ensure a robust software product**
      - **2.A.1: Software Assurance processes provide reduced risks and higher confidence in SW products**
        - **2.A.1.A: Assess the software level and criticality and determine SA program risks**
          - SAS-024: Perform and keep up to date SSCA
          - SAS-032: Contents of SA Plan
            - SAS-032a: A description of all planned assurance activities
            - SAS-032b: Risks associated with any activities required by this Standard that have not been planned and funded
            - SAS-032c: Results of Software Assurance Characteristics Assessment (SACA)
            - SAS-032d: Initial safety criticality assessment results
          - SAS-045: Generate, report and maintain quality records on SA Project activities
            - SAS-045a: Independent Assessment of SW Classification as per NPR 7150.2
            - SAS-045b: SA Characteristics Assessments (SACA)
            - SAS-045c: SW Safety Criticality Assessment (SSCA)
            - SAS-045d: Any Waivers or Deviations to meeting this Standard
            - SAS-045f: SA Standard Compliance Matrix at SA level (See appropriate compliance matrix in SAEHB)
          - SAS-087: Verify software severity levels assigned and maintained
          - SAS-088: Assure risk associated with DR/PR is entered into risk system(s)
          - SAS-094: Assure SW change will not compromise Project's risk posture and is compliant with NPR 7150.2
          - SAS-101: Conduct independent SSCA
        - **2.A.1.B: Provide a planned, maintained, comprehensive, working software assurance process**
          - SAS-019: Assess and concur on Provider's SA plan

⋮ ⋮

      - **2.A.3: Software interim and final products conform to project needs and requirements**
        - **2.A.3.A: Assure software final and interim products are of sufficient quality for the project**
          - SAS-054: Assure planned development and maintenance processes are followed
          - SAS-065: Report and track to closure SW problems, findings, risks and SW safety issues, elevating as necessary
          - SAS-078: Assure regression testing is planned, documented and covers SW changes
          - SAS-089: Evaluate and concur completion of verification and validation plans and procedures
          - SAS-097: Assure CM was used and correct version and accompaniments are delivered
          - SAS-071: Assure decisions and analyses are documented
          - SAS-083: Assess accept as is resolutions
          - SAS-084: Assess change resolution addressed the problem and does not introduced new ones
          - SAS-108: Collect track and trend SW quality metrics and SW defects
          - SAS-109: Perform reliability analyses
          - SAS-117: Assure SW quality metrics include reliability indicators
    - **2.B: Identify and resolve faults throughout the development process in a timely manner**
      - **2.B.1: Faults, defects, or other issues have been found and resolved as part of the development process**
        - **2.B.1.A: Track, address, and trend issues via a closed loop problem resolution process**
          - SAS-063: Independently verify and track closure of action items from reviews
          - SAS-065: Report and track to closure SW problems, findings, risks and SW safety issues, elevating as necessary
          - SAS-080: Review resolutions of SW related defects
          - SAS-083: Assess accept as is resolutions

Figure 9. Fragment of textual listing of the mapping for Objective (2) of the SOH, of Figure 5

A compact graphical presentation of the same portion of the SOH-draft NSAS mapping is seen in Figure 10, where yellow squares alongside the SOH 2.#.#.# strategies hold just the numeric portion of the SAS-### requirements numbers. It is obvious from this that multiple of the requirements contribute evidence towards each of the strategy node—this is as might be expected given the broad ranging nature of the 2.#.#.# branch of the SOH (most especially for *2.A.1.B Provide a planned, maintained, comprehensive, working software assurance process*—the heart of the SOH).

2.A.1.A Assess the software level and criticality and determine SA program risks	024	032a	032b	032c	032d	045a	045b	045c	045d	045f	087	088	094	101
2.A.1.B Provide a planned, maintained, comprehensive, working software assurance process	019	021	025	039	030	031	035	036	032a	032e	032f	032g	032h	032i
	032j	032k	032l	032m	032n	032o	032p	037	045g	045h	056	047a	047b	047c
	047d	047e	047f	047g	049a	049h	101							
2.A.2.A Assure use of a maintained, comprehensive working software development process	018	022	038	050	051	052	055	060	061	070	076	077	078	049a
	049c	049d	049f	049g	049h	049i	072	075a	075c	075d	075e			
2.A.2.B Assure software development processes are followed and do not introduce errors	053	054	057	060	063	064	065	068	069	070	082	089	090	091
	092	093	095	096	071	083	084	085	086	097	098	100		
2.A.3.A Assure software final and interim products are of sufficient quality for the project	054	065	078	089	097	071	083	084	108	109	117			
2.B.1.A Track, address, and trend issues via a closed loop problem resolution process	063	065	080	083	084	085	086							
2.B.2.A Assure risks are collected, analyzed, tracked, and addressed	037	052	057	058	059	065	068	074d	074e	074h	074n	075b	075c	077
	079	096	075f	088										

Figure 10. A compact graphical presentation of the SOH 2.#.#.# strategy nodes and alongside each, the NSAS-### numbers of the draft NSAS requirements assessed as providing evidence towards that node

6 Conclusion

We have described the RISC concept and illustrated its potential use through application to the BioSentinel project. In particular, we have created several argument fragments, based on our earlier mapping of BioSentinel V&V artifacts to the SOH, that show how software assurance claims in the SOH are explicated, refined, and ultimately met by V&V evidence items.

There is a natural relation between the SOH and RISC concepts, in that objectives hierarchies can provide the skeletal core of the arguments that form the central artifacts of a RISC. There is also a close connection between scenario development through an integrated safety analysis, representation of those scenarios as BTDs, and the correspondence between those BTDs and the arguments. Since we did not develop BTDs in this report, we will not consider this further. Moreover, we have not considered all aspects of RISCs, in particular the evaluation of alternatives according to the ASARP principle. A proper treatment of this important concept would require an expansion of our scope beyond safety to consider other system properties as well as appropriate notions of cost.

Finally, the next step of this work, based on our understanding and interpretation of SOHs, RISCs, and their relationship, is to develop tool requirements to support their development. We intend to do this in the context of an existing assurance case tool, AdvoCATE, which we used to create the structured arguments in this report.

Acknowledgments

This research was carried out at NASA Ames Research Center, and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, sponsored by the NASA Office of Safety and Mission Assurance. At JPL this activity was managed through the Assurance and Technology Program Office as part of the NASA OSMA Software Assurance Research Program.

References

[1] NASA, Office of Safety and Mission Assurance, Software Assurance Working Group. Software Assurance Objectives Hierarchy. [Online]. Available: https://sma.nasa.gov/docs/default-source/sma-disciplines-and-programs/software-assurance/sa-hierarchy.pdf

[2] NASA System Safety Steering Group (S3G), *NASA/SP-2010-580, NASA System Safety Handbook. Volume 1, System Safety Framework and Concepts for Implementation*, NASA/SP-2010-580, NASA, Nov. 2011. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120003291.pdf

[3] ——, *NASA/SP-2014-612, NASA System Safety Handbook. Volume 2: System Safety Concepts, Guidelines and Implementation Examples*, NASA/SP-2010-580, NASA, Nov. 2011. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150015500.pdf

[4] P. Castle, E. Denney, M. Feather, and G. Pai, "Applicability of the Software Assurance Objectives Hierarchy to BioSentinel," SARP / Adapting the Software Assurance Objectives Hierarchy to a Model-based Reuse Process / Deliverable 1, April 2018.

[5] NASA, Office of Safety and Mission Assurance. NASA-STD-8739.8, Software Assurance Standard (Baseline w/ Ch 1 of 5/5/05). [Online]. Available: https://standards.nasa.gov/standard/nasa/nasa-std-87398

[6] UK Ministry of Defence (MOD), *Safety Management Requirements for Defence Systems*, Defence Standard 00-56, Issue 4, 2007.

[7] The Assurance Case Working Group (ACWG), "Goal Structuring Notation Community Standard Version 2," Jan. 2018. [Online]. Available: https://scsc.uk/r141B:1

[8] E. Denney and G. Pai, "Towards a formal basis for modular safety cases," in *34th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2015)*, Sep. 2015.

[9] ——, "Automating the Assembly of Aviation Safety Cases," *IEEE Transactions on Reliability*, vol. 63, no. 4, pp. 830–849, 2014.

[10] ——, "A methodology for the development of assurance arguments for unmanned aircraft systems," in *Proceedings of the 33rd International System Safety Conference (ISSC)*, Aug. 2015.

[11] ——, "Architecting a Safety Case for UAS Flight Operations," in *34th International System Safety Conference (ISSC)*, Aug. 2016.

[12] ——, "Tool support for assurance case development," *Journal of Automated Software Engineering*, vol. 25, no. 3, pp. 435–499, Sep. 2018.

[13] E. Denney, G. Pai, and I. Whiteside, "Modeling the Safety Architecture of UAS Flight Operations," in *Computer Safety, Reliability, and Security. SAFECOMP 2017.*, ser. Lecture Notes in Computer Science, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds. Springer, Cham, Sep. 2017, vol. 10488.

[14] R. Clothier, E. Denney, and G. Pai, "Making a Risk Informed Safety Case for Small Unmanned Aircraft System Operations," in *Proceedings of the 17th AIAA Aviation Technology, Integration, and Operations Conference (ATIO 2017), AIAA Aviation Forum*, no. (AIAA 2017-3275), Jun. 2017.

[15] NASA, Office of the Chief Engineer. NPR 7150.2B, NASA Software Engineering Requirements. [Online]. Available: https://standards.nasa.gov/standard/nasadir/npr-71502

[16] NASA. NASA-STD-8719.13, NASA Software Safety Standard. [Online]. Available: https://standards.nasa.gov/standard/nasa/nasa-std-871913

A Acronyms

ARC Ames Research Center
CDR Critical Design Review
FMEA failure modes and effects analysis
FRR Flight Readiness Review
FSW Flight Software
FMP Fault Management Plan
GSN Goal Structuring Notation
ICD Interface Control Document
IRR Integration Readiness Review
IV&V Independent V&V
KDP Key Decision Point
NSAS NASA Software Assurance Standard
NSER NASA Software Engineering Requirements
PDR Preliminary Design Review
PTSR Preliminary Technical and Safety Review
RISC Risk Informed Safety Case
SAP Software Assurance Plan
SRP Software Release Plan
SDV Simulink Design Verifier
SEMP Systems Engineering Management Plan
SDP Software Development Plan
SLS Space Launch System
S&MAP Safety and Mission Assurance Plan
SOH Software Assurance Objectives Hierarchy
SQA Software Quality Assurance
TVAC Thermal Vacuum
TVPM Thermal Vacuum Power Management
V&V Verification and Validation
WSIM Workstation Simulator
BTD Bow Tie Diagram
ISA integrated safety analysis
ASARP as safe as is reasonably practicable