

The Essential Synthesis of Problem Frames and Assurance Cases

Elisabeth A. Strunk John C. Knight

Department of Computer Science, University of Virginia
151 Engineer's Way, Charlottesville, VA 22904-4740
{strunk | knight}@cs.virginia.edu

Abstract

Problem frames and assurance cases are two current research areas that can improve—and have improved—system dependability, in critical and non-critical systems alike. While these two techniques are effective separately, their synthesis is much more powerful. This paper describes the rationale behind the synthesis, the particular pieces that influence each other, and the beginning of a process to integrate the two in software system development.

1. Introduction

Both problem frames [3] and assurance cases [5] support the development and assurance of dependable software. Problem frames were originally created, and are primarily used, to help developers elicit and structure software system requirements. They support software assurance by clarifying requirements and their system context in a way that enables developers to make rigorous arguments for system validity in the context of the system's problem. Assurance cases were developed to provide a means of documenting an argument that a system possesses a specific dependability attribute. The most common use of assurance cases at present is in the provision of arguments for system safety.

These are two important technologies that appear to be related only marginally, if at all. Each makes a valuable contribution to system dependability, but we claim that a carefully formulated combination of the two provides a value over and above their individual contributions. This increased value arises because each technology enhances the other.

The combination enhances assurance cases by providing additional structure and rigor. The current formulation of assurance cases leaves many aspects of the argument excessively informal. This degree of informality is, in fact, a weakness because it increases the possibility that the assurance argument will be flawed.

A suitable formulation of certain parts of the system using problem frames enables substantial rigor to be used in the informal elements of assurance cases with which we are concerned.

The combination enhances problem frames by providing a rigorous statement establishing satisfaction of the frame concern. Showing that the frame concern has been addressed is imprecise in the current formulation of problem frames, and the necessary precision and completeness are supplied by the basic structure of assurance cases.

With these two strong ties between assurance cases and problem frames, we have concluded that the two technologies are, in fact, strongly related. This relationship is such that the synthesis of the two is an extremely valuable structure. In this paper we discuss the relationship between problem frames and assurance cases, and we show how a composite of the two provides considerable benefit. We begin by presenting some background information on assurance cases. We then discuss how problem frames can enhance assurance cases and vice versa. Finally, we briefly describe a software development approach that exploits the integration of the two, and present our conclusions.

2. Assurance Case Background

Assurance cases are the state of the art in rigorous, but non-formal, dependability argumentation. The most common use of assurance cases at the moment occurs in the documentation of safety, and safety cases have been built for a variety of production systems. Graphical notations have been designed to enable the documentation of assurance cases in a manner that is easy for humans to understand and that can be manipulated by machine. The most widely used of these notations is the Goal Structuring Notation (GSN) [5]. The argument in Figure 1 illustrates the use of GSN in a simple safety case.

In general, a safety case, an example of an assurance case, is “a documented body of evidence that provides

a convincing and valid argument that a system is adequately safe for a given application in a given environment” [1]. In its simplest form, it contains an instance of each of three essential elements: (1) a safety goal or claim; (2) evidence that the goal has been satisfied; and (3) an argument linking the evidence to the goal in a way that leads one to believe that the goal is justified by the evidence. This basic structure is applied recursively to produce, for real systems, a hierarchic structure with the overall goal for the system at the root. The overall goal is decomposed into more specific sub-goals, with evidence supporting the argument included where the goals are specific enough to accommodate it.

An assurance goal is typically a nonfunctional requirement such as a dependability requirement. Functional requirements tend not to be considered in an assurance case unless they relate to other, nonfunctional requirements. We believe that the emphasis on separating functional and nonfunctional requirements weakens the overall assurance argument, and that problem frames are the most promising approach to integrating these various system goals.

3. Goals and Context of the Assurance Case

The overall quality of both developed systems and assurance cases is limited by the quality of their stated requirements. For systems, those requirements include both the informal statement of what the system is to do

and the formal model of the system set out in its specification. For assurance cases, the requirements define the top-level goal that the assurance case must show.

Because of their role in helping to document and structure requirements, problem frames can contribute significantly to the state of the art in the assurance case. Specifically, problem frames can: (1) assist in the creation of the initial goal of the assurance case; and (2) help to structure the assurance case context.

3.1 Assurance Case Goals

Typically, the top-level goal of an assurance case is a crucial dependability requirement of the system to be assured. Safety and security are the most common goals; the goal as stated might be something like “the system is acceptably safe to operate.” What safety means for the system, and what level of safety is acceptable, is part of the context for that goal.

We believe that it is more effective to state the goal of the assurance case in terms of the problem to be solved than to state it in terms of the developed system (the solution to the problem). Dependability concerns, such as the reliability concern (for which there already exists a discussion [3]), can be used to express the dependability properties that might normally be separated into their own assurance case. By ensuring that the problem, including all of its functional and dependability concerns, is addressed, we can assure the specific dependability concerns in a more robust way.

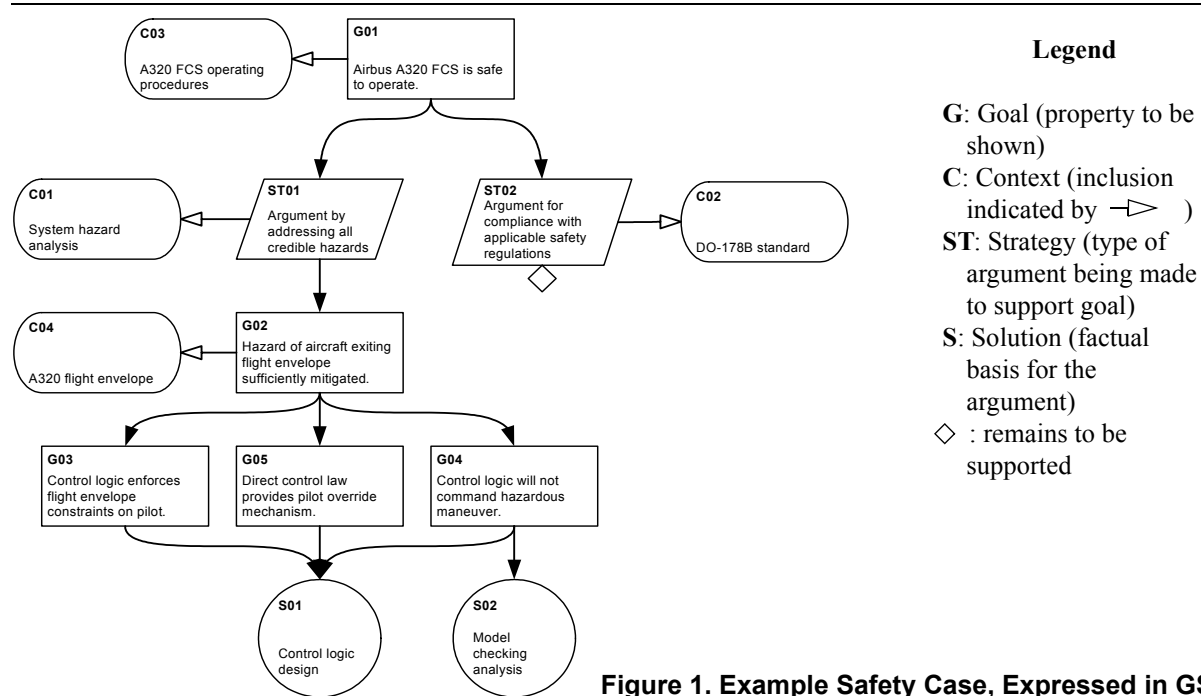


Figure 1. Example Safety Case, Expressed in GSN

The new top-level assurance goal, then, becomes “the problem that is addressed by the system is solved.” When elaborated using problem frames, this top goal is split into two subgoals. The first goal is that the frame for the problem is a valid representation of the problem. The second is that the frame concern is addressed, i.e., that the machine solves the problem. Further subgoals about correctness of implementation will branch from this second subgoal.

3.2 Assurance Case Context

There is an absence in the assurance case literature of a systematic method to capture context necessary to support the assurance argument. Context enters into an assurance case implicitly via assumptions, definitions of domain terminology, strategies for decomposing arguments, and justifications for certain types of inferences. Without explicit acknowledgement, it is difficult to understand the basis for contextual aspects of the argument or to evaluate whether the warrants based on context are reasonable.

Although GSN represents context explicitly, as a first-class node in an assurance case graph, its treatment of context is rudimentary. Context nodes may refer to external documents such as domain standards, definitions, or any other domain-related facts, so their meaning is vague. Moreover, the contribution of a context node to an argument, such as which assumptions it enables the argument to make, is typically unstated and left to the reader to judge. Finally, unlike context diagrams, GSN does not provide support for expressing relationships among contextual elements. Consequently, the reader of an assurance case may have difficulty in discerning the contribution of context to an assurance case and in evaluating the validity of the contextual basis of an argument.

Basing the assurance case goal on the problem frame introduces a way to more elegantly and efficiently structure the assurance case context. There are two reasons why this is the case. First, the context diagram can identify how the various domain and standards documents that are linked to the goal apply to the system. The diagram conveys the domains with which the system will interact, enabling the assurance case to document which domains the system influences directly and which domains it ultimately influences but over which it does not have direct control. The structural depth of the diagram allows: (1) relationships to be clearly depicted; (2) domain-specific terms to be documented with other concepts in their particular domains; (3) references to other sources to be included together with the context pieces necessary to under-

stand those sources; and (4) the distinctions between direct and indirect influence to be made.

The second reason is that the structure and decomposition of the context can be reflected in the structure and decomposition of the frame. Each machine is connected to certain domains, and it is within those domains that the context lies. The information captured in the context diagram can be linked to the design choices made in a system—and thus to the assurance arguments that are made about that system—through the structure provided by the frame. Without this structure, evaluation of the system with respect to its context is much more difficult. Since the assurance case is an informal argument that the system solves this problem, structuring context in a comprehensible way and clearly linking it to the parts of the system to which it is relevant is essential.

4. Assuring Satisfaction of the Frame Concern

To the best of our knowledge, a general way to determine whether the frame concern is satisfied for any particular system has not been completely researched. Using software architecture to elaborate and support this argument has been studied [2], but this is a general method for incorporating specific techniques, each of which in turn must be shown to support the frame concern in a specific way.

Because the frame concern is stated in terms of the problem domain, it is necessarily informal. Thus, an argument that it has been addressed must also be informal (although it may have formal elements). Assurance cases were developed specifically to support informal argument in a rigorous way. Thus, we believe that assurance cases are the best method for constructing the overall frame concern argument.

Assurance cases do not constrain the arguments that can be made about the frame concern. One potential argument, for example, would be that the frame concern is modified or addressed by the software architecture used in the system’s design. This argument should include either a justification that the architecture fully satisfied the frame concern or a strategy for decomposing the frame concern that shows what portion of it is addressed by the architecture. The strategy or justification would include an *informal* description of the architecture, enabling a developer to see the interpretation of the architecture in terms of the problem it addresses and thus decide whether the architecture did indeed address the problem.

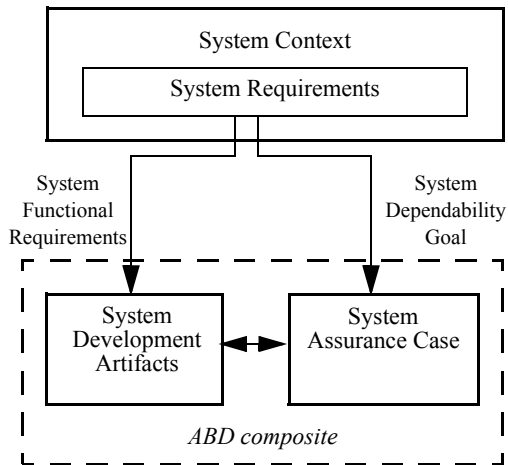


Figure 2. Assurance-Based Development

5. Assurance-Based Development

The synthesis of assurance cases and problem frames leads to a new way of looking at software system development. In current practice, a system is usually developed before its assurance case. Others have suggested that the two be developed in a roughly parallel fashion [4], but the synthesis implies that they should be developed together explicitly and tied very closely.

We are studying a new method, called *assurance-based development* (ABD), in which the separate parts of the typical construction of an assured system—requirements and context analysis, system development, and assurance case creation—are integrated. Integration enables the whole system development process to benefit from the careful thought given to system dependability, thought that is required to create the system’s assurance case.

The major components of assurance-based development and their high-level interactions are shown in Figure 2. At the center of the technique are the *system assurance case* and the *system development artifacts*. Because we emphasize the integration of these two components, we call the combination of the two the *ABD composite*. The two components are developed in parallel, and their development is coordinated to ensure that assurance goals and development artifacts are coupled explicitly and systematically. The coupling will thus reveal the evidence that the assurance case needs the development artifact to provide.

Shown at the top of the figure are components labeled *system context* and *system requirements* with the former enclosing the latter. The context in which a system operates influences the system requirements in many ways. The system requirements are used by both

the system assurance case and the system development artifacts. The system requirements include the dependability requirements such as safety, and thus determine the primary goal of the assurance case. The system requirements also include the functional requirements, and so they are the starting point for the development lifecycle. In assurance-based development, determining the system context accurately and fully is a major element.

The synthesis of problem frames and assurance cases forms the core of ABD. Problem frames provide the basic structuring mechanism for the requirements of the system, and also for its context. The assurance case goals and context are based on the system’s problem frame requirements. The development process carries the problem from the frame through to its software solution, arguing rigorously that the solution satisfies the problem, both in validity and verifiability.

6. Conclusion

While problem frames and assurance cases are effective separately, their synthesis is much more powerful. We have described the rationale behind the synthesis, the particular pieces that influence each other, and the beginning of a process to integrate the two in software system development.

References

- [1] Bishop, Peter and Robin Bloomfield. “A Methodology for Safety Case Development.” <<http://www.adelard.co.uk/resources/papers/index.htm>>
- [2] J. Hall, M. Jackson, R. Laney, B. Nuseibeh, and L. Rapanotti. “Relating Software Requirements and Architectures using Problem Frames.” Proceedings of IEEE International Requirements Engineering Conference (RE’02), Essen, Germany, September 2002.
- [3] Jackson, M. “Problem Frames.” Addison-Wesley: Harlow, England, 2001.
- [4] Kelly, T. P. “A Systematic Approach to Safety Case Management.” Proc. of SAE 2004 World Congress, Detroit, MI, March 2004.
- [5] Weaver, R. A. and T. P. Kelly. “The Goal Structuring Notation - A Safety Argument Notation.” Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004.

Acknowledgments

We appreciate William Greenwell’s assistance with the assurance case material presented here and his contribution of the safety case in Figure 1. This work was funded in part by NSF grant CCR-0205447 and NASA grant NAG1-02103.