Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-28 (2016) 025–030

Maintaining Safety Arguments via Automatic Allocation of Safety Requirements

Ioannis Sorokos, Yiannis Papadopoulos, Leonardo Bottaci

Department of Computer Science, University of Hull, Hull, HU67RX, UK

(e-mail: I.Sorokos@2012.hull.ac.uk, Y.I.Papadopoulos@hull.ac.uk, L.Bottaci@hull.ac.uk)

Abstract: The 'safety case' documents the safety argument developers of safety-critical systems employ to convince of their systems' safety, in compliance with safety standard regulation and advice. Despite the considerable body of knowledge that has evolved, constructing and maintaining a safety case remains a significant challenge. Especially for contemporary systems, due to their scale and complexity, safety cases can grow to require hundreds of pages of documentation. In this paper, we propose a method which aims to address these concerns. In numerous safety standards, such as the aerospace ARP4754-A, the concept of Development Assurance Levels (DALs) is used to control the safety assessment process and influence the safety case. Our method is based on automatically constructing a safety argument from an annotated system architecture model. To perform this construction, we employ previous work towards automatically allocating DALs to such a model and combining it with an appropriate safety argument pattern. The method is enabled through the state-of-the-art model-based dependability tool, HiP-HOPS. The advantage of this approach is that when the design changes, the impact of changes can be automatically reflected in the structure of a re-synthesised safety argument for the system.

Keywords: safety case maintenance; automation; safety requirements; ARP4754-A; DAL decomposition.

1. INTRODUCTION

Contemporary assurance of safety-critical systems often involves the production and maintenance of the safety case. The safety case should present 'a clear, convincing and comprehensive' argument that the subject system is acceptably safe (Kelly, 2004). Graphical notations such as the Goal Structuring Notation (GSN) (Kelly, 1998) and the Claims-Arguments-Evidence (CAE) (Bishop, et al., 2004) notation have provided many tools for improving the representation of such arguments. Despite these advances, the process of constructing and maintaining safety cases remains largely a manual one. This is particularly the case in the early to interim stages of development, where the system's design experiences numerous iterations. During these critical stages, safety case developers need to manage significant amounts of information and construct arguments, the complexity of which are comparable to the scale and complexity of the underlying system. In (Denney, et al., 2013, p. 1) a preliminary safety case for surveillance of airport surfaces (EOSAN, 2011) is quoted to be 'about 200 pages' and 'expected to grow as the operational safety case is created'. It is safe to assume that emerging and future technologies will only further exacerbate this issue, introducing more complexity and interactions whose operation needs to be accounted for in a relevant safety case.

The contribution of this paper is a method which automates the construction and maintenance of a part of the safety case. The method focuses on the production of a (partial) preliminary safety argument for civil aircraft, where the applicable safety standard is ARP4754-A. The standard employs a process known as the 'Development Assurance Process', whereby Development Assurance Levels (DALs) are assigned hierarchically across the system architecture in a top-down approach. These levels prescribe the rigor with which safety assessment procedures are to be applied accordingly throughout the relevant sections of the system. Previous research has demonstrated that it is feasible to optimally allocate DALs, based on a cost estimation of implementing a component for a given DAL (Sorokos, et al., 2015). The approach makes use of the Hierarchically Performed Hazard and Origins Propagation Studies (HiP-HOPS) tool (Papadopoulos, et al., 2011).

The method presented here expands on this notion to produce a safety case fragment from this allocation which can form the basis of a preliminary safety case. Previous work towards automating construction of safety cases in (Basir, et al., 2008) and (Denney, et al., 2013) focused on generating safety cases for automatically generated code based on formal software safety certification. An approach comparable to ours can be seen in (Sljivo, et al., 2015). Although at first glance similar, there are considerable

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.11.005

differences. Their approach constructs safety arguments from 'safety contracts', specifications of necessary safety properties for particular commercial-off-the-shelf software components. The safety contracts are generated from model of the subject software specified in Fault Propagation and Transformation Calculus (FPTC). Our approach is applicable from the early stages of design, requiring less rigorous annotation of system architecture models instead of formal methods for software components. It can be combined freely with other approaches for automatic or traditional manual production of safety cases. We should note that the concept of DALs is shared throughout numerous standards, commonly referred to as Safety Integrity Levels (SILs). Thus, the method presented here can be generalized across such standards as the domain-neutral IEC 61508 and the automotive-domain ISO 26262 which use SILs as well.

The following section introduces the development of safety cases and the ARP4754-A standard. The method of automatic allocation of DALs will also be briefly presented, as it provides the backbone of the safety case construction. In Section 3, the method for producing the safety argument is presented. In Section 4, the method is applied to a simplified model. The model is then modified and the method re-applied to illustrate the benefits of automation. The final section includes a discussion of the method's implications together with relevant and further work.

2. BACKGROUND

2.1. Introduction to Safety Argument Notation

Safety argument notations such as the Goal Structuring Notation (GSN) and the Claims Arguments Evidence (CAE) notation (also known as the Adelard Safety Case Development/ASCAD notation) were introduced to provide a structured approach to constructing and representing safety arguments. In both notations, graphical shapes depict elements considered fundamental to structuring any given safety case. The philosophy behind the identification of these elements is largely attributed to the work of philosopher Stephen Toulmin on the analysis of practical argumentation (Toulmin, et al., 1984). These elements are the safety case's claims/goals, arguments/strategies and evidence/solutions in CAE/GSN respectively. Claims describe attributes, objectives or constraints that the underlying system is argued to achieve. Evidence is factual information which supports the truthfulness of the claims made. Arguments connect claims to evidence by explicating the rationale which links them. As both of these notational systems have developed, additional concepts have been introduced. Of particular interest are GSN's argument patterns and parameterized arguments. Safety argument patterns are inspired by the concept of architectural and software design patterns found in (Alexander, et al., 1977) and (Gamma, et al., 1994). Argument patterns attempt to capture good practice in safety case design by "abstracting the argument strategy from the details of a particular argument" (Hawkins & Kelly, 2013, p. 3).

2.2. Civil Aircraft Safety Assessment.

The framework for producing the evidence necessary for safety assurance is typically provided through the guidelines of one or more safety standards relevant to the subject system. Depending on the industry, the system may need to be certified by independent authorities and/or official regulatory bodies. The safety standard ARP4754-A and its supporting documents provide guidelines for developing systems compliant with regulations for civil aircraft development. As mentioned in the introduction, the safety assessment process the ARP4754-A advocates, is centralized around the concept of Development Assurance Levels (DALs) (SAE, 2010, pp. 22-23), commonly known in other standards as Safety Integrity Levels (SILs). These levels are assigned to subsystems and components of the system's architecture and encapsulate the rigor with which safety assessment activities are conducted. Both the regulations and the standard itself advocate conducting safety assessment in a top-down process, mirroring the evolution of the system's development.

The ARP4754-A places the safety assurance processes in the context of the 'v-model' of system development (SAE, 2010, p. 24). Under this v-model, the early stages of system development involve the identification and analysis of the system's functionality, i.e. the system's functions. A Functional Hazard Analysis (FHA) is recommended to elicit the hazards associated with the failure modes of each function (SAE, 2010, p. 31). Each hazard is classified into one of five categories of severity, ranging from No Effect, Minor through to Catastrophic. Depending on the hazard in question, mitigation measures, system requirements or design choices might also be considered.

At this point, development proceeds to identifying the system architecture responsible for implementing each function. Once this relationship has been established, a qualitative failure analysis, such as Fault Tree Analysis (FTA) (Vesely, et al., 1981), is conducted to determine if and how failures in the underlying systems can trigger a hazardous functional failure. Through FTA, the minimum combinations of system failures which are necessary and sufficient to cause a function to fail can be determined. These are commonly known as 'minimum cut sets' and in the ARP4754-A are referred to as Functional Failure Sets (FFSs) (SAE, 2010, p. 11). This is a recursive process; when sub-systems supporting each system are introduced, FTA or an equivalent analysis will be conducted to determine their failure contribution to the system's and a