Facilitating Reuse of Certification Artefacts Using Safety Contracts

PhD Student: Irfan Sljivo
Main Advisor: Hans Hansson
Co-Advisors: Jan Carlson, Barbara Gallina
PhD Start and Expected Defence Dates: May, 2012 – June, 2017

Mälardalen Real-Time Research Centre, Mälardalen University,
Västerås, Sweden
{irfan.sljivo, hans.hansson, jan.carlson, barbara.gallina}@mdh.se

Abstract. Safety-critical systems usually need to be certified according a domain-specific safety standard. To reduce the cost and time needed to achieve the safety certification, reuse of certification artefacts together with the corresponding safety-relevant software components is needed. The certification artefacts include safety claims/goals and the supporting evidence that are documented in a safety case. The safety case is required by some safety standards to show that the system is acceptably safe to operate in a given context. Assumption/guarantee contracts can be used to capture the dependencies between a component, including its certification artefacts, and a particular operating context.
In this paper we present a research proposal on how safety contracts can be used to facilitate systematic reuse of certification-relevant artefacts. More specifically, we explore in which way should such contracts be specified, how can they be derived, and in which way can they be utilised for reuse of safety case argument-fragments and the artefacts those arguments include.

1 Introduction

Safety-critical systems are those that can result in harm or loss of human life, or damage to the environment. A trend in safety-critical systems is that new functionalities are added mainly through software, which explains why a modern car has from 70 to 100 embedded computers on board, with overall software that scales up to 100 million lines of code[1]. To ensure that these safety-critical software-intensive systems achieve sufficient levels of safety, most of such systems must be certified according to a set of domain-specific safety standards. The cost of certifying these systems is estimated at 25-75% of the development costs [16], with the cost of producing the verification artefacts for highly critical applications reaching up to 1000 USD per code line [7]. In most cases, as a part of certification an additional time- and money- consuming activity of providing

[1] see http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code

a safety case is required. A safety case is presented in form of an explained and well-founded structured argument to clearly communicate that the system is acceptably safe to operate in a given context [11]. The goal of the safety argument is to connect safety claims about the system with the supporting evidence.

More and more safety standards are offering support for reuse to reduce the production costs and time needed to achieve certification. For example, in the latest versions of both airborne (DO178-C) and automotive (ISO 26262) industry standards techniques for reuse are explicitly supported through the notions of Reusable Software Components (RSC) within airborne [1] and Safety Elements out of Context (SEooC) within automotive industry [10]. Different approaches can be used to facilitate software reuse by making it more structured and systematic. For example, Component-based Development (CBD) is the most commonly used approach to achieve reuse within aerospace industry [17]. As a part of CBD approaches for safety-critical systems, contract-based approaches have received significant attention for some time already [4, 5, 6, 8]. A contract for a component is defined as an assumption/guarantee pair, where the component offers guarantees about its behaviour under the assumptions about its environment. While such CBD approaches have been successfully used to support reuse of software components, they lack support for reuse of certification-relevant artefacts such as safety arguments and the supporting evidence.

Reusing safety-relevant properties, presented as safety claims in the safety argument, and the supporting evidence is hindered by the fact that safety is a system property, and that certification is usually performed with respect to the specific system. To achieve reuse of the safety claims about a component, the dependencies between the claims and the corresponding system-specific properties should be identified and captured. The captured dependencies should ensure that the safety claim and the supporting evidence are reused only when they actually hold. Moreover, since what is safety-relevant in one system does not have to be safety-relevant in another system, there is need for identifying which claims and evidence are relevant for the system in which the component is reused. In this thesis we focus on developing the notion of component safety contracts that capture safety-relevant properties of components developed and prepared for safety certification independently of the system in which they will be used. More specifically, we investigate how and in which way should such contracts be specified to support systematic reuse of certification-relevant artefacts. We assume reuse of software components without changes to the component. In case of adaptations of the reused component for a particular context, its safety contracts and the corresponding certification-relevant artefacts need to be revisited. Confidence that all relevant contract assumptions have been identified should be established in each context in which the component is reused. If needed, contracts should be enriched with additional assumptions to establish the confidence.

The paper is organised as follows: In Section 2 we present the research methodology and the identified research goals and questions. We present the corresponding research contributions achieved to date and the evaluation plans in Section 3. We present the related work in Section 4.

2 Research Description

2.1 Research Methodology

We start the research by reviewing the state of the art, which leads to identification of the research problem and definition of a concrete research goal. To continue the research, the concrete research goal is divided into subgoals that are further refined into research questions. For each of the questions we study the literature and propose a solution that we evaluate on a simple example. After sufficient results have been achieved for each of the defined research questions, a validity of the overall approach is demonstrated on an industrial case study.

2.2 Problem Statement and Research Questions

As mentioned in Section 1, safety-critical systems industries have a problem with high production costs and the time needed to achieve safety certification for the software-intensive systems. One way of addressing this issue is by enabling reuse of not only software components, but the accompanying certification-relevant artefacts as well. The overall goal of the thesis is: *to facilitate reuse of certification-relevant artefacts related to the software components being reused.*

The certification-relevant artefacts include safety-relevant properties and the supporting evidence that are presented in a safety argument. Capturing the safety-relevant properties can be achieved by the safety contracts, as mentioned in Section 1. Just as the components need to be designed for reuse, so do the contracts as well. To achieve the overall goal we first need to establish a contract formalism to define how the contracts should be specified such that they provide support for systematic reuse of the captured safety-relevant properties. To address this issue we define the first research question: *How should the safety contracts for software components be specified in order to facilitate systematic reuse of certification-relevant artefacts?*

Safety-critical systems are characterised by a wide-range of properties on which the guaranteed safety behaviour of components depends, hence it is challenging to derive contracts with a complete set of assumptions on the environment under which the contract guarantees hold. While many works deal with contract formalisms and how the contracts should look like, the issue of their derivation has not been sufficiently addressed. To address this challenge we define the second research question: *How can valid component safety contracts be derived from the results of different types of failure analyses to support systematic reuse of certification-relevant artefacts?*

The safety-relevant properties specified in the contracts are usually supported by evidence in form of different safety analyses. As mentioned in Section 1, the safety-relevant properties captured within contracts are presented as safety claims within the safety argument together with the supporting evidence. While the argument presents the safety-relevant information in a comprehensible way, safety contracts capture the safety-relevant information in a more rigorous manner. The fact that both, the safety argument about a component and the component safety contracts, deal with the same information makes the contracts an

important aid in facilitating reuse of the argument-fragments and supporting evidence. To address these issues we define the third research question: *How can the component safety contracts be used to facilitate systematic reuse of the argument-fragments and supporting evidence?*

3 Research Contributions and Evaluation Plans

In this section we present the research contributions achieved so far, and the plans for evaluation of the overall approach of our research.

3.1 Research Contributions

Strong and Weak Contract Formalism. This contribution addresses the first research question. We present our extension of the contract formalism to include specification of strong and weak contracts and the related reasoning [12, 14]. Typically, behaviours of software components can be captured in assumption/guarantee contracts in such way that the component guarantees its behaviour if the stated assumptions on its environment are met. Strong contracts capture properties that should hold in all systems of intended use of the component, while the weak contacts capture system-specific properties.

Derivation of Safety Contracts from Failure Analyses. This contribution addresses the second research question. We present methods for deriving safety contracts from failure logic analyses such as Failure Propagation and Transformation Calculus (FPTC) [15] and Fault Tree Analysis (FTA). Just as hazard analysis is the basis for safety engineering at the system level, derivation of contracts and identification of related assumptions plays a similar role at component level. We use well-established failure logic analyses recommended by safety standards as the basis for contract derivation and assumptions identification.

Contribution 3: A Method for Reuse of Safety Case Argument-fragments and Supporting Evidence This contribution addresses the third research question. We present a method to generate reusable argument-fragments from compositional safety analysis. We first focus on the generation of context-specific safety case argument-fragments from safety contracts for components developed out-of-context [13]. To develop a sound argument we define certain aspects that should be covered in an argument-fragment created using contracts, and provide set of rules that perform the generation of such argument-fragments for specific contexts. Next, we provide methods for derivation of safety contracts from failure analyses and reuse of the generated argument-fragments and the supporting evidence [15]. Once the initial contracts have been derived for a component developed out-of-context, the contracts are further supported by evidence such as failure analyses reports and additional V&V evidence. When the component is instantiated in a particular system, artefacts relevant for achieving safety goals of the particular system are identified through satisfied contracts and used in generation of the context-specific argument-fragment.

3.2 Evaluation Plans

To demonstrate specification of safety contracts, their derivation and usage for argument-fragment and evidence reuse, an industrial safety-critical subsystem/component is needed. The component should be required to comply with the corresponding domain-specific safety standard and it should qualify to be developed outside of the system in which it will be used in form of SEooC or RSC. An example of such component can be found in product-line scenarios where a component is developed independently of a specific product and then used in different product-line products. In our works so far we used two systems for demonstrating the approaches, Scania's fuel level estimation system for trucks and busses [13] and an aircraft wheel-braking system [14, 15]. Besides the small examples used to demonstrate individual contributions, we plan to use a larger industrial case study for evaluating the overall approach. Our plans assume cooperation with Volvo Construction Equipment (VCE). VCE has a broad range of 150 different construction machines divided into 7 product families. Many of their vehicles have a lifting arm and a corresponding Lifting Arm Control Unit (LACU). The goal is to develop the LACU according to ISO 26262 using our approach to facilitate systematic reuse of certification-relevant artefacts. The evaluation should uncover the efficiency of our approach in reducing the costs related to certification of LACU for all the different construction machines in which LACU is used.

4 Related Work

A range of formal contract-based approaches can be found in recent related work [4, 5, 6, 8]. Except for partial support through introduction of strong and weak assumptions [8], these works do not provide support for capturing safety-relevant information for reuse. Unlike in our work, none of these works actually focuses on how the contracts should be specified and used for reuse of software components together with their accompanying certification-relevant artefacts. Moreover, these works do not address the issue of contract derivation.

There has been many works on automating generation of safety arguments from different artefacts [2, 3, 9]. While these approaches offer a way to speed up creation of safety arguments, they do not provide support for components that are developed out-of-context and have different artefacts that are valid for different systems. Once such components are instantiated in a particular context, their corresponding argument should contain only information relevant for that particular context. We use contracts to identify those information and artefacts that are relevant for the particular context in which the component is reused.

Acknowledgements

This work is supported by the Swedish Foundation for Strategic Research (SSF) via project SYNOPSIS as well as EU and Vinnova via the Artemis project SafeCer.

References

[1] AC 20-148. *Reusable Software Components*. Federal Aviation Administration, Dec. 2004.

[2] E. Armengaud. Automated safety case compilation for product-based argumentation. In *Embedded Real Time Software and Systems (ERTS)*, Feb. 2014.

[3] N. Basir, E. Denney, and B. Fischer. Building heterogeneous safety cases for automatically generated code. In *Infotech@ Aerospace Conference*. AIAA, 2011.

[4] S. S. Bauer, A. David, R. Hennicker, K. Guldstrand Larsen, A. Legay, U. Nyman, and A. Wasowski. Moving from specifications to contracts in component-based design. In *15th international conference on Fundamental Approaches to Software Engineering*, FASE'12, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] I. Ben-Hafaiedh, S. Graf, and S. Quinton. Reasoning about safety and progress using contracts. In *12th international conference on Formal engineering methods and software engineering*, ICFEM'10, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis. Multiple viewpoint contract-based specification and design. In *Formal Methods for Components and Objects: 6th International Symposium (FMCO)*, 2007.

[7] R. Bloomfield, J. Cazin, D. Craigen, N. Juristo, E. Kesseler, J. Voas, et al. Validation, Verification and Certification of Embedded Systems. Technical report, NATO, Oct. 2005.

[8] W. Damm, H. Hungar, B. Josko, T. Peikenkamp, and I. Stierand. Using contract-based component specifications for virtual integration testing and architecture design. In *Design, Automation & Test in Europe Conference & Exhibition (DATE 2011)*. IEEE, 2011.

[9] E. Denney and G. Pai. A lightweight methodology for safety case assembly. In *31st International Conference on Computer Safety, Reliability and Security (*SafeComp*)*. Springer-Verlag, Sept. 2012.

[10] ISO 26262-10. *Road vehicles — Functional safety — Part 10: Guideline on ISO 26262*. International Organization for Standardization, 2011.

[11] T. P. Kelly. *Arguing Safety — A Systematic Approach to Managing Safety Cases*. PhD thesis, University of York, York, UK, Sept. 1998.

[12] I. Sljivo, J. Carlson, B. Gallina, and H. Hansson. Fostering Reuse within Safety-critical Component-based Systems through Fine-grained Contracts. In *International Workshop on Critical Software Component Reusability and Certification across Domains (CSC)*, June 2013.

[13] I. Sljivo, B. Gallina, J. Carlson, and Hansson. Generation of Safety Case Argument-Fragments from Safety Contracts. In *33rd International Conference on Computer Safety, Reliability, and Security (*SafeComp*)*. Springer, Sept. 2014.

[14] I. Sljivo, B. Gallina, J. Carlson, and H. Hansson. Strong and weak contract formalism for third-party component reuse. In *3rd International Workshop on Software Certification (*WoSoCer*)*. IEEE, Nov. 2013.

[15] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri. A method to generate reusable safety case fragments from compositional safety analysis. In *14th International Conference on Software Reuse (ICSR)*. Springer-Verlag, January 2015.

[16] N. R. Storey. *Safety Critical Computer Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[17] J. Varnell-Sarjeant, A. A. Andrews, and A. Stefik. Comparing Reuse Strategies: An Empirical Evaluation of Developer Views. In *8th International Workshop on Quality Oriented Reuse of Software (QUORS)*. IEEE, July 2014.