Building Multiple-Viewpoint Assurance Cases Using Assumption/Guarantee Contracts

Irfan Sljivo
Mälardalen Real-Time Research Centre
Mälardalen University
Västerås, Sweden
irfan.sljivo@mdh.se

Barbara Gallina
Mälardalen Real-Time Research Centre
Mälardalen University
Västerås, Sweden
barbara.gallina@mdh.se

ABSTRACT

Assurance cases in form of structured arguments are often required by standards to show that a system is acceptable for its intended purpose with respect to a particular assurance viewpoint such as safety or security. The goal of such a case is to present an argument that connects the requirements of a particular viewpoint with the supporting evidence. Building a set of assurance cases for the different viewpoints can be time-consuming and costly. Means are needed to automate and reuse the assurance case artefacts between the assurance cases for the different viewpoints.

In this paper we present how assumption/guarantee contracts can be used to facilitate reuse of assurance case artefacts by building multiple-viewpoint assurance cases from the contracts. More specifically, we build upon the previous work on argument-fragment generation from such contracts to allow for generating viewpoint specific argument-fragments. We illustrate the approach on a motivating case.

1. INTRODUCTION

The hierarchy of safety being over security is changing and more and more evidence is presented that equal efforts need to be invested on achieving both properties [8]. Hence many critical systems such as passenger vehicles require certificates for the different properties such as safety and security. Assurance cases as a way to document the assurance process in form of an argument have been used to assure system safety and security [3]. There has been many discussions on the interplay of the different properties that require assurance, mainly safety and security, and how their interconnection can be handled in the corresponding assurance processes. These discussions are still ongoing for instance in the automotive domain community (e.g., [13]) and suggest including security assurance aspects in the next edition of the automotive functional safety standard ISO 26262 [7]. The discussions in the avionics domain have resulted in publishing the RTCA DO-326A [12] standard, which focuses on the security aspects that may affect aircraft airworthiness. The initiatives in both domains aim at expanding the safety assurance case with only those security assurance aspects deemed safety-relevant. To reduce the efforts needed to build such overlapping assurance cases, means are needed to reuse assurance artefacts and reasoning between the different assurance cases.

Assumption/guarantee contracts have been used to achieve reuse of safety artefacts together with reusable safety-relevant components [17]. A contract of a component is defined as a pair of assumptions and guarantees, where the component offers guarantees about its own behaviour provided that the environment meets the assumptions. Since a component can be described with a set of such contracts, the different contracts can be used to represent the different viewpoints of the component behaviour [2]. For example, a functional viewpoint can be represented with a contract stating functional behaviour of the component related to its interface, while a timing viewpoint can be represented with contracts addressing the timing behaviour of the component. The contract theory [2] allows for the composition of the different viewpoints and ensures their consistency.

In our previous work [15, 17] we have demonstrated how the assumption/guarantee contracts supported by evidence can be used to generate argument-fragments, which can be used to compose the safety assurance case. In this work we present how associating assurance viewpoints with such contracts can be used to build multiple-viewpoint assurance case arguments by generating argument-fragments for the different assurance viewpoints. Generating such viewpoint-specific argument-fragments from the same source supports reuse of assurance artefacts related to the contracts that belong to the different viewpoints. More specifically, we propose a contract-based assurance meta-model as an extension of the previous work [17], then we map the contract elements to the argumentation notation elements and extend the argument-generation algorithm to generate viewpoint-specific argument-fragments. We use a fictitious avionics wheel-braking example from the avionics standard [18] to illustrate the approach.

The rest of the paper is structured as follows: In Section 2 we provide some essential background information. We present the overall idea in Section 3 and the motivating case in Section 4. We present the related work in Section 5 and finally, we present the conclusions and future work in Section 6.

2. BACKGROUND

In this section we provide background information on assurance cases and a contract-based approach that facilitates generating assurance case argument-fragments.

2.1 Assurance cases

An **assurance case** is defined as *"a collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy the particular requirements."* [10]. An assurance case is a generic term for cases where an argument is used to connect the requirements with the supporting evidence to assure certain property of the system such as safety and security. The essential part of the assurance case is the *argument*, which is usually divided to a product and a process part. The process part relates to the conformance to the mandated assurance process, and the product part details the specific measures taken to make the product exhibit the required property. The argument can be represented in different ways ranging from free text to more formal notations. To facilitate creation of a well-structured and clear argument, the graphical argumentation notation – Goal Structuring Notation (GSN) [1], is proposed. GSN can be used to record and present the main elements of any argument. The main purpose of GSN is to show how **goals** (claims about the system depicted with rectangles), are broken down into **subgoals** and supported by **solutions** (the gathered evidence depicted with circles used to support the goals). Moreover, **away goals** are used to refer to goals developed in other argument modules. A subset of GSN elements is shown in Figure 1. To provide better portability and exchange of the assurance arguments, Structured Assurance Case Meta-model (SACM) [10] is presented as a standard for representing structured assurance cases. Different assurance case argument modelling tools support importing and exporting of assurance cases in SACM-compliant format.

2.2 Contract-based assurance

A traditional component *contract* $C = \langle A, G \rangle$ is composed of *assumptions* (A) on the environment of the component and *guarantees* (G) that are offered by the component if the assumptions are met [2]. Since not all contracts are required to hold in every system, the notion of *strong and weak contracts* [14] is introduced to distinguish between the strong contracts whose assumptions should be always satisfied, and the weak contracts whose guarantees (H) are offered only when together with the strong assumptions the corresponding weak assumptions (B) are also satisfied. While the traditional contracts are developed with the primary goal of formal verification, strong and weak contracts are introduced as a methodological extension to provide support for utilising the behaviours of reusable components for the development of assurance cases. Since the reusable components are usually developed with different configuration parameters that can be used to tailor the behaviour of the component for the different usage contexts, the traditional contracts and the strong/weak contracts lack support for making strong assumptions (the ones that must be met) for the different contexts. *Configuration-aware contracts* [16] are introduced to explicitly distinguish between the component configuration parameters that have constant value within a single context and the operational variables whose value is specified within a predefined range. OCRA[1] (Othello Contracts Refinement Analysis) is a tool for checking refinement between contracts. The Othello constraints syntax that we use in our examples includes both boolean and temporal logic operators for specifying the assumptions and guarantees of the contracts.

Figure 1: A subset of GSN elements

Safety contracts are a specific types of contracts that deal specifically with component behaviours that are deemed relevant from the perspective of hazard analysis. The Safety Element out-of-context Metamodel (SEooCMM) [17] is developed around the notion of safety contracts and deals with the safety viewpoint of the system. SEooCMM provides the base for evidence reuse through product-based argument-fragment generation. It captures properties of an out-of-context component, composed of safety-contracts, evidence and the assumed safety requirements. Each safety requirement is satisfied by at least one safety contract, and each contract can be supported by one or more evidence. Furthermore, both contracts and evidence can be further clarified with informal context statements. Figure 2 shows the core elements of SEooCMM in solid borders.

The rules for the generation of the argument-fragments from safety contracts [15] generate an argument-fragment by first arguing that all allocated requirements on the component are satisfied. Then the rules iterate for each of the requirements and generate a sub-argument that all the contracts supporting the requirement are satisfied. The algorithm for the generation of the contract-satisfaction argument-fragments is achieved through a model-to-model (M2M) transformation as defined in Op-

[1]https://ocra.fbk.eu/

erational QVTo [4]. The algorithm for the transformation from SEooCMM to SACM [17] generates an argument-fragment for each satisfied contract by showing that the contract assumptions are met by the environment, and that the contract is sufficiently complete by attaching the supporting evidence.

3. MULTIPLE VIEWPOINT ASSURANCE USING CONTRACTS

As mentioned in Section 1, the overlapping of the assurance processes for the different assurance viewpoints such as safety and security requires mechanisms to consider their interplay in both the system and the assurance case domain. Contracts represent a way to analyse and ensure coexistence of the different viewpoints in the system domain. In this section we present how contracts can also be used to consider the interplay of the different assurance viewpoints, namely safety and security, in the assurance case domain.

3.1 Multiple-Viewpoint SEooCMM

The SEooCMM component meta-model is limited to a single viewpoint and one type of contracts. To consider multiple viewpoints and different contract implementations, we generalise and extend SEooCMM to provide the basis for a generic *Multiple-Viewpoint (*) Safety Element out-of-Context Meta-Model* (*SEooCMM) presented in Figure 2. As mentioned in Section 2.1, the assurance case is requirements oriented. Hence, we define in *SEooCMM an *assurance viewpoint* as a set of requirements. Since each requirement can be satisfied by one or more contracts, and each of the contracts can be supported by different evidence, through these connections captured in *SEooCMM, we can extract those relevant to a particular viewpoint. Since a single contract can be used to satisfy (fully or partially) requirements that belong to different viewpoints, the corresponding evidence and the argument-fragment related to those contracts can be reused between the different assurance case viewpoints. Moreover, *SEooCMM allows that a single requirement can belong to one or more viewpoints, which also facilitates reuse of the argument-fragments supporting the satisfaction of the requirement in the different assurance case viewpoints.

Since *SEooCMM deals with multiple viewpoints, as well as out-of-context components and their transition to a particular context, we do not focus only on the strong and weak contract as the out-of-context and in-context component meta-models can have different implementations of the notion of a contract. Instead, we include the different contract implementations in *SEooCMM to facilitate generation of assurance case argument-fragments regardless of the specific contract type used to specify the behaviour of components.

3.2 *SEooCMM-based argument-fragments

Although the different contract types facilitate capturing different aspects of the component behaviours, in general, their assumptions and guarantees can be mapped equally to the argumentation elements. Essentially, the two main components of a contract, its assumptions and guarantees, are mapped to argumentation elements by: (1) creating goals for each guarantee to show how the guarantee is satisfied, and (2) by providing away goals for each assumption to show how each of the assumptions is satisfied by a contract in the environment. For both of the cases, the actual assumption and guarantee statements could be provided as context elements outside of the goals for better clarity of the argument [15]. An exception to this mapping are the configuration assumptions. Since they can have only one value per context that cannot change, instead of arguing over such assumptions, we provide them only as contextual statements. The contracts element mapping to the argumentation elements is shown in Table 1.

Table 1: Conceptual mapping of the different contracts to the corresponding GSN elements

GSN-elements	Contract elements
Goal	All properties representing guarantees (strong, weak, traditional and configuration-aware)
Away goal	Properties representing assumptions (strong, weak, traditional, and operation)
Context	Configuration assumptions

Since the mapping between the different contracts elements and the argumentation elements is generally the same, the resulting arguments follow the same structure. But due to the different granularity of the assumptions and guarantees in the different contracts, the level of detail of the generated argument-fragments for the different types of contracts differs. For example, the weak contracts are usually captured as special types of implications within guarantees of traditional contracts. Hence, to achieve an argument with the same level of detail as the argument based on the strong and weak contracts, the traditional contract should be first parsed to identify the suitable implications, and check that the left side of the implication is true (i.e., that the implicit assumption is satisfied), in order to obtain an argument with the same level of detail as the one generated from the strong and weak contracts.

Unlike the strong/weak and traditional contracts, the configuration-aware contracts explicitly distinguish between configurable parameters and operational variables. On the one hand, the configuration assumptions deal with configuration parameters that do not change but are fixed in a particular context, hence we consider that satisfaction of such assumptions does not require to be argued over, but just presented for information purposes. On the other hand, the assumptions on the operational variables are of more interest to argue over, as their value can change during operational time. Due to these characteristics, the configuration-aware contracts

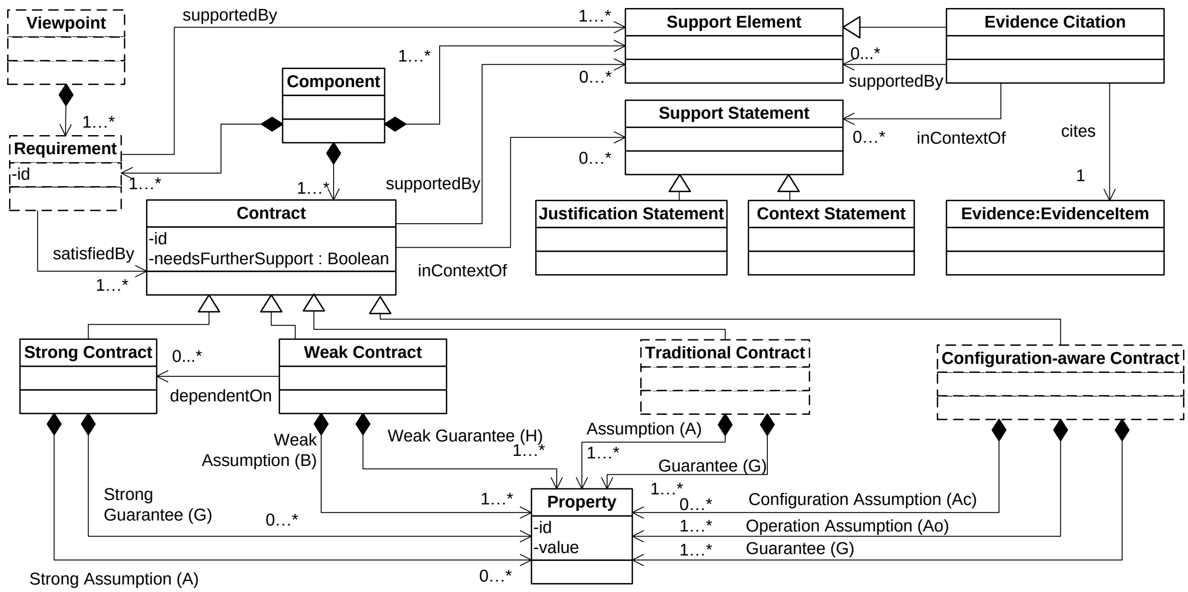

Figure 2: SEooCMM [17] extension to *SEooCMM (the dashed elements newly added)

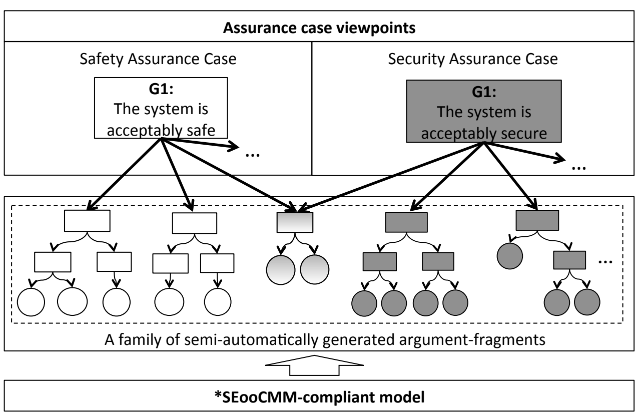

Figure 3: Building multiple viewpoint assurance cases with *SEooCMM

with the two level of assumptions offer purging of the argument from the irrelevant assumptions and guarantees by filtering only those relevant for the particular system described by the set of parameters.

3.3 From contracts to assurance viewpoints

Assuring different viewpoints of a system requires that specific analyses are performed and evidence gathered to identify and support the different viewpoint-specific requirements. Instead of running separate assurance processes, the idea with contract-based multi-assurance approach is to use contracts as the meeting point between the different assurance viewpoint processes. Handling of the interplay of the e.g., safety and security in the system domain within the contracts allows us to automate and reuse within the creation of the viewpoint-specific assurance arguments.

*SEooCMM allows that different viewpoints can have some requirements in common, as well as that one contract can be used to support several requirements. Similarly, an evidence item such as hazard or vulnerability analysis, can be used to support several contracts. By capturing this interplay between the different viewpoints in *SEooCMM, we can identify the viewpoint-specific requirements, their supporting contracts and the corresponding supporting evidence in the contracts. Since this information represents the basis of an assurance case argument, we utilise and adapt the existing argument-fragment generation techniques mentioned in Section 2.2 to semi-automatically generate argument-fragments discussed in Section 3.2. The set of such argument-fragments can be used to build a viewpoint-specific product-based argument assuring e.g., that the system is acceptably safe or secure (Figure 3). For clarity we depict security related argument-fragments in solid grey colour, the safety related ones in white and the shared ones in the white-grey gradient.

The transformation from SEooCMM to SACM [17] adapted to *SEooCMM is shown in Algorithm 1. To generate the viewpoint-specific argument-fragments we build upon the generation rules and the M2M algorithm described in Section 2.2. The Algorithm 1 considers only the requirements of a specific viewpoint, instead of all the allocated requirements to a component. That way distinct arguments can be generated for the different viewpoints, while portions of those arguments related to the common entities are automatically identified and included in the resulting assurance case arguments. Furthermore, in Algorithm 1 we parametrise the rules for the higher-level argument generation to handle the different viewpoints, while the lower-level arguments follow pre-established argument patterns. To support argument-fragment generation from different

Algorithm 1 M2M Transformation from *SEooCMM to GSN SACM argumentation meta-model

```
*SEooCMM2SACM(in *SEooCMM, in Viewpoint, out SACM){
  topClaim(in *SEooCMM::Viewpoint, out SACM::GSN_Goal);
  for each *SEooCMM::Requirement req do
    if req belongs to Viewpoint then
      for each *SEooCMM::Contract sc that satisfies req do
        sc2claim(in *SEooCMM::Contract, out SACM::GSN_Goal);
        scCont(in *SEooCMM::Context, out SACM::GSN_Context);
        scJust(in *SEooCMM::Justification, out SACM::GSN_Justification);
        addSubGoals(in *SEooCMM::Contract, out SACM::GSN_Goal);
        sc2context(in *SEooCMM::Contract, out SACM::GSN_Context);
        if sc type is traditional or strong/weak then
          for each Assumption a in sc do
            if a is satisfied then
              a2claim(in *SEooCMM::Contract, out SACM::GSN_Goal);
              away2a (in *SEooCMM::Contract, out SACM::GSN_AwayGoal);
            else if a not satisfied then
              a2cEvid (in *SEooCMM::Contract, out SACM::GSN_CounterEvidence);
            end if
          end for
        else if sc type is configuration-aware then
          confA2context(in *SEooCMM::Contract, out SACM::GSN_Context);
          for each satisfied operational Assumption opA in sc do
            if opA is satisfied then
              opA2claim(in *SEooCMM::Contract, out SACM::GSN_Goal);
              away2opA (in *SEooCMM::Contract, out SACM::GSN_AwayGoal);
            else if opA not satisfied then
              opA2cEvid (in *SEooCMM::Contract, out SACM::GSN_CounterEvidence);
            end if
          end for
        end if
        for each *SEooCMM::EvidenceCitation ec supporting sc do
          ec2claim(in *SEooCMM::EvidenceCitation, out SACM::GSN_Goal);
          ecSol(in *SEooCMM::EvidenceItem, out SACM::GSN_Solution);
          ecCont(in *SEooCMM::Context, out SACM::GSN_Context);
          ecJust(in *SEooCMM::Justification, out SACM::GSN_Justification);
        end for
      end for
    end if
  end for

}
```

contract implementations, we parametrise the part of Algorithm 1 to consider the exception related to the configuration assumptions of the configuration-aware contracts, as discussed in 3.2.

4. MOTIVATING CASE

In this section we illustrate how *SEooCMM-based contract specification can be used to generate assurance viewpoint specific arguments. More specifically, we use the airplane Wheel Braking System (WBS) example and perform a partial safety and security analyses. We specify a set of contracts and generate assurance viewpoint specific argument-fragments.

4.1 Wheel Braking System

The WBS information is adapted from the Avionics Recommended Practices ARP4761 [18] and a previous work [14]. WBS takes two input brake pedal signals and outputs the brake signal that is applied on the wheel. The system is designed with two redundant Brake System Control Units (subBSCUs), where both perform the same calculations. By default the value of the primary subBSCU is forwarded to the output if its validity flag is true, while the validity flag for the entire BSCU is calculated based on the validity flags of both subBSCUs and the *Guard* component. The command values and the validity flag are forwarded to the hydraulics component, where either the command is used if the validity flag is true, or the alternate mode is engaged.

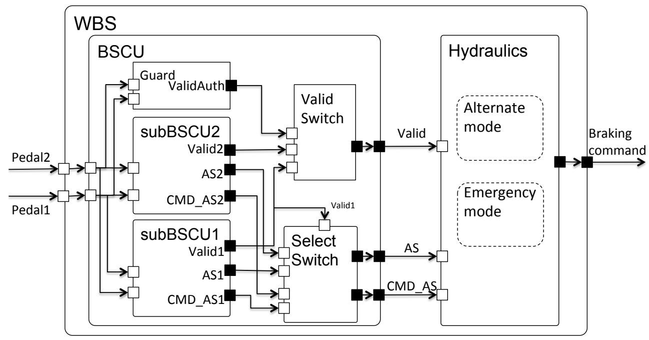

Figure 4: Wheel Braking System Architecture

The systems safety analysis revealed several hazards. We focus on the hazard *H1: inadvertent braking*, where WBS may issue a braking command when not supposed to. The redundant inputs and subBSCUs are used to prevent that a wrong braking calculation is issued to the hydraulics component.

The systems security analysis revealed a security vulnerability *V1: unauthorised braking*, where false pedal signals that may be sent over the communication bus may trigger unauthorised braking, which represents both safety and security risk at the same time. To ensure that unauthorised messages do not trigger the braking system, a security kernel *Guard* is used to ensure that the pedal signals are received from the authorised components.The WBS architecture with the guard component is shown in Figure 4.

4.2 Application example

In the example we address the following two requirements allocated to BSCU: (1) the software safety requirement *SwSafR1*: *"BSCU shall not issue a valid braking command without present input."*; and a software security requirement *SwSecR1*: *"BSCU shall not issue a valid braking command in presence of unauthorised input signals."*. The BSCU as the main software controller of WBS is developed to address these requirements.

The contracts supporting these requirements allocated to BSCU and its subcomponents are shown in Tables 2 and 3. The *P1_key_send* and *P2_key_send* flags in the Guard and BSCU contracts represent assumptions that the Pedal1 and Pedal2 inputs contain key and sender information. These assumptions are a prerequisite to establish whether the origin of the signals Pedal1 and Pedal2 can be authenticated (represented with *P1_auth* and *P2_auth* flags). While the BSCU-1 and Guard-1

Table 2: An example of a BSCU contract

$\mathbf{A}_{BSCU-1}$: $\mathbf{G}_{BSCU-1}$:	*Pedal1==Pedal2* AND *P1_key_send* AND *P2_key_send*; (*P1_auth* AND *P2_auth* AND noDubleFault) **implies** (Valid AND CMD);
$\mathbf{C}_{BSCU-1}$:	noDubleFault entails that BSCU can handle failure of a single subBSCU;
$\mathbf{E}_{BSCU-1}$:	*name*: BSCU integration testing results;

Table 3: A subset of the subBSCU, Guard and ValidSwitch contracts

$\mathbf{A}_{subBSCUx-1}$: $\mathbf{G}_{subBSCUx-1}$:	*Pedal1==Pedal2*; noDubleFault **implies** Validx;
$\mathbf{C}_{subBSCUx-1}$:	Validx output can be trusted in presence of at most one internal fault;
$\mathbf{E}_{subBSCUx-1}$:	*name*: subBSCU unit testing results;
$\mathbf{A}_{Guard-1}$: $\mathbf{G}_{Guard-1}$:	*P1_key_send* AND *P2_key_send*; (*P1_auth* AND *P2_auth*) **implies** ValidAuth;
$\mathbf{E}_{Guard-1}$:	*name*: Guard unit testing results;
$\mathbf{A}_{ValidSwitch-1}$: $\mathbf{G}_{ValidSwitch-1}$:	-; ((Valid1 or Valid2) AND ValidAuth) **implies** Valid;
$\mathbf{C}_{ValidSwitch-1}$:	Valid is true if the inputs are authenticated and at least one subBSCU returns valid result;
$\mathbf{E}_{ValidSwitch-1}$:	*name*: Switch unit testing results;

contracts support both of the requirements, contracts subBSCUx-1, subBSCUx-2 and ValidSwitch-1 support only the SwSafR1 requirement. Based on these dependencies we apply the Algorithm 1 and generate the different assurance case viewpoints. The current version of the algorithm is not implemented yet, hence we apply the algorithm manually. Figure 5 shows the two assurance case viewpoints merged for space limitations, and following the same colouring scheme as Figure 3.

5. RELATED WORK

The interplay between safety and security as the prime examples of the assurance viewpoints has motivated many research works [11]. Lautieri et al. [9] explore the commonalities between safety and security assurance and propose a common methodology for the certification of highly modular safe and secure systems. The methodology aims at combining the safety and security analyses to reuse the same evidence for both authorities.

Dobbing [5] proposes dependability-by-contract (DbC) approach that aims at building a single dependability case that argues the achievement of an acceptably safe and secure system. The contracts in DbC are extended beyond assumptions and guarantees to include some information needed for building a dependability case.

Gallina et al. [6] focus on the reuse of process elements between the safety and security certification via the Security-informed Safety-oriented Process Line Engineering (SiSoPLE) approach. The approach first performs identification of commonalities and variabilities between the two processes and builds a common process line from which a specific process can be engineered to satisfy different security or safety standards.

In this work, we propose to further extend the contract-based approach by considering the different assurance viewpoints in the underlying component meta-model to allow for reuse of assurance artefacts and generation of product-based argument-fragments specific to a particular assurance viewpoint. While creating a single dependability case is beneficial, certification bodies usually require a case that is specific to the assurance viewpoint of the corresponding standard. Hence, we introduce *SEooCMM to allow for extraction of such assurance viewpoint-specific information.

6. CONCLUSIONS AND FUTURE WORK

Managing the interplay between the different assurance viewpoints such as safety and security is becoming an important issue. Due to the proliferation of standards and the risk of duplication of work and even mitigation measures, the corresponding standards are starting to consider their interplay. Achieving a unified assurance process for the different viewpoints based on the different standards is not realistic. Furthermore, unifying assurance case arguments leads to overloading them with unneeded information. Hence, means are needed to reduce the time and efforts needed to build the viewpoint-specific assurance case arguments. Component contracts in component-based software engineering provide means to capture the interplay of the different system viewpoints. In this work we propose that such contracts can also represent means to manage the interplay of the different assurance viewpoints when building the multiple-viewpoint assurance cases. We have proposed a component meta-model *SEooCMM to capture the relationship between the different viewpoints, requirements, contracts and the evidence, all aspects needed to automate generation of product-based assurance case arguments.

As our future work, we plan to investigate how contracts can be used to capture the interplay between the different viewpoints from the corresponding standards perspective. Moreover, we plan to explore how the contract-based assurance approach can be integrated in the different domain-specific development processes. Finally, we plan to evaluate the usefulness of the proposed approach on a real-world use case.

Acknowledgements

This work is supported by the EU and VINNOVA via the ECSEL JU project AMASS (No 692474).

7. REFERENCES

[1] GSN Community Standard Version 1. Technical report, Origin Consulting (York) Limited, November 2011.

[2] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis. Multiple Viewpoint Contract-Based Specification

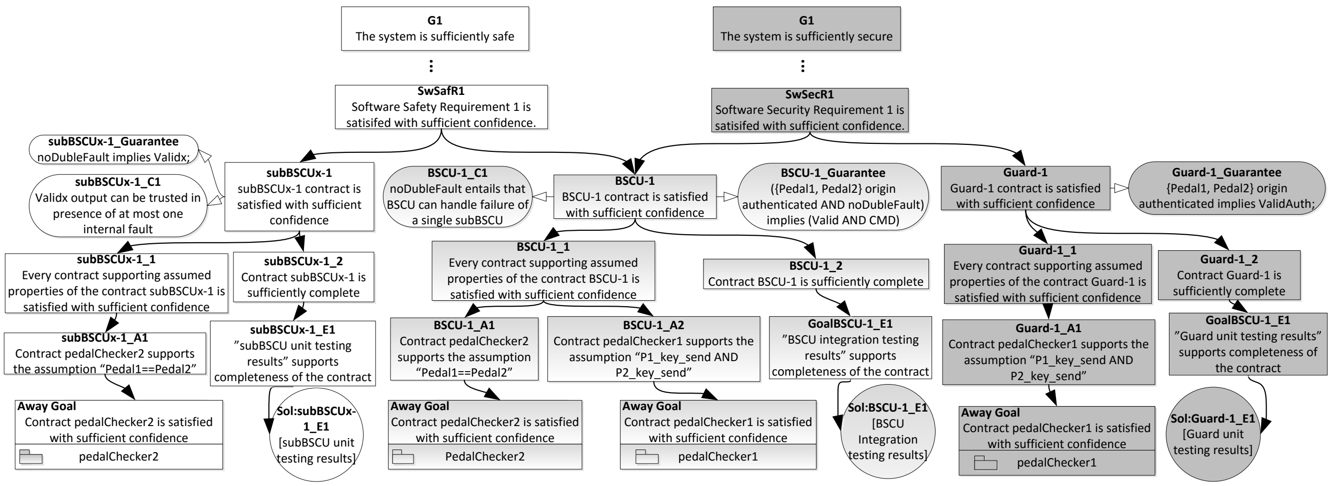

Figure 5: A snippet of the two assurance case viewpoints with a shared argument-fragment

and Design. In *Formal Methods for Components and Objects*, volume 5382 of *LCNS*, pages 200–225. Springer, 2007.

[3] R. Bloomfield and P. Bishop. Safety and assurance cases: Past, present and possible future–an Adelard perspective. In *Making Systems Safer*, pages 51–67. Springer, 2010.

[4] S. Boyko, R. Dvorak, and A. Igdalov. The art of model transformation with operational qvt, 2009.

[5] B. Dobbing and S. Lautieri. Dependability by contract. In *15th Safety-critical Systems Symposium*, pages 35–51. Springer, February 2007.

[6] B. Gallina and L. Fabre. Benefits of security-informed safety-oriented process line engineering. In *34th Digital Avionics Systems Conference (DASC)*, pages 8C1–1–8C1–9, September 2015.

[7] International Organization for Standardization (ISO). *ISO 26262: Road vehicles — Functional safety*. ISO, 2011.

[8] N. Kuntze, C. Rudolph, G. B. Brisbois, M. Boggess, B. Endicott-Popovsky, and S. Leivesley. Security vs. safety: Why do people die despite good safety? In *Integrated Communication, Navigation and Surveillance Conference*, pages 1–10, Apr. 2015.

[9] S. Lautieri, D. Cooper, and D. Jackson. SafSec: Commonalities between safety and security assurance. In *Constituents of Modern System safety Thinking*, pages 65–75. Springer, 2005.

[10] O. M. G. (OMG). SACM: Structured Assurance Case Metamodel. Technical report, Version 1.1, OMG. http://www.omg.org/spec/SACM, 2015.

[11] L. Piètre-Cambacédès and M. Bouissou. Cross-fertilization between safety and security engineering. *Rel. Eng. & Sys. Safety*, pages 110–126, 2013.

[12] Radio Technical Commission for Aeronautics (RTCA). *DO-326A: Airworthiness Security Process Specification*. RTCA DO-326A, 2014.

[13] C. Schmittner, Z. Ma, and T. Gruber. Standardization challenges for safety and security of connected, automated and intelligent vehicles. In *Int. Conference on Connected Vehicles and Expo*, pages 941–942, November 2014.

[14] I. Sljivo, B. Gallina, J. Carlson, and H. Hansson. Strong and Weak Contract Formalism for Third-Party Component Reuse. In *3rd Int. Workshop on Software Certification*, pages 359–364. IEEE, November 2013.

[15] I. Sljivo, B. Gallina, J. Carlson, and H. Hansson. Generation of Safety Case Argument-Fragments from Safety Contracts. In *33rd Int. Conference on Computer Safety, Reliability, and Security*, volume 8666 of *LCNS*, pages 170–185. Springer, September 2014.

[16] I. Sljivo, B. Gallina, J. Carlson, and H. Hansson. Configuration-aware Contracts. In *4th Int. Workshop on Assurance Cases for Software intensive Systems*. Springer, September 2016.

[17] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri. A method to generate reusable safety case argument-fragments from compositional safety analysis. *Journal of Systems and Software: Special Issue on Software Reuse*, July 2016.

[18] Society of Automotive Engineers. *ARP-4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*. SAE, 1996.