# Generation of Safety Case Argument-Fragments from Safety Contracts

Irfan Sljivo, Barbara Gallina, Jan Carlson, and Hans Hansson

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

**Abstract.** Composable safety certification envisions reuse of safety case argument-fragments together with safety-relevant components in order to reduce the cost and time needed to achieve certification. The argument-fragments could cover safety aspects relevant for different contexts in which the component can be used. Creating argument-fragments for the out-of-context components is time-consuming and currently no satisfying approach exists to facilitate their automatic generation. In this paper we propose an approach based on (semi-)automatic generation of argument-fragments from assumption/guarantee safety contracts. We use the contracts to capture the safety claims related to the component, including supporting evidence. We provide an overview of the argument-fragment architecture and rules for automatic generation, including their application in an illustrative example. The proposed approach enables safety engineers to focus on increasing the confidence in the knowledge about the system, rather than documenting a safety case.

**Keywords:** Safety Case Argument-fragments, (Semi-)automatic Generation, Safety Contracts, Composable Certification, Out-of-context Components

## 1 Introduction

The cost for achieving certification is estimated at 25-75% of the development costs [16]. As a part of certification, a safety case in form of a structured argument is often required to show that the system is acceptably safe to operate. To reduce cost and time-to-market, more and more safety standards are offering support for reuse within safety cases. Safety Element out of Context (SEooC) is an example of a concept for reuse proposed by the automotive ISO 26262 standard [12]. Building on such reusable elements, an approach to composable certification has been proposed [5]. The approach aims at achieving incremental certification by composing reusable argument-fragments related to safety elements, whose behaviour is specified through safety contracts. We define argument-fragments as parts of the system safety argument that argue about safety aspects relevant for the individual components.

In our previous work [15] we developed a safety contract formalism to facilitate reuse of components developed out-of-context. The safety contracts capture safety-relevant behaviours of the components in assumption/guarantee pairs. The semantics of such a pair is that if the assumption holds then the guarantee will also hold. The assumption/guarantee pairs are characterised as being either strong or weak. The strong contract assumptions are required to be satisfied in all contexts in which the component is used, hence the strong guarantees are offered in every context in which the component can be used. On the other hand, the weak contract guarantees are only offered in the contexts in which the component can be used and that satisfy the corresponding weak assumptions.

The strong and weak contracts allow us to distinguish between properties that hold for all contexts and those that are context-specific. Since every context has specific safety requirements, argument-fragments for out-of-context components may partially cover safety aspects relevant for several contexts. Creating argument-fragments for components developed out-of-context is a time-consuming activity. (Semi-)automatic generation of such argument-fragments from safety contracts would speed up the activity and allow for generation of context-specific argument-fragments. Moreover, the safety engineers would have the possibility to focus on increasing the confidence in the knowledge about the system, rather then on clerical tasks such as documenting a safety case [13].

Currently, no satisfying approach exists that facilitates generation of argument-fragments for out-of-context components. The main contribution of this paper is that we propose such an approach, capable to (semi)automatically generate argument-fragments from safety contracts and related safety requirements and evidence. As the basis for our approach we developed a meta-model that captures relationships between the safety contracts, safety requirements and evidence. To support the generation of argument-fragments from the safety contracts we provide conceptual mapping between the meta-model and argumentation notation elements. To perform the generation we provide the resulting argument-fragment architecture and a set of rules to generate the argument-fragments.

We demonstrate our approach on a Fuel Level Estimation System (FLES) and its variants that are used within Scania's trucks and busses. We focus on a single component of FLES that estimates the fuel level in the tank. This component represents a good candidate to be developed as SEooC as it is used with slight variations in many different variants. We use the safety contracts not only to capture the knowledge we have about the behaviour of the component, but also the evidence supporting the guaranteed behaviour. Moreover, by connecting in-context safety requirements with the weak safety contracts that address the requirements, we enable only those safety properties of the component relevant for the particular context to be used when developing the argument-fragment. This allows us to support more efficient creation of the argument-fragments as well as generation of context-specific arguments that contain information relevant for the context in which the component is used.

Compared to existing works, we focus on generation of argument-fragments for components developed and prepared for safety certification independently of the system in which they will be used. Approaches to generating safety case arguments [9, 3] usually extract the necessary information to build an argument from artefacts provided to satisfy some process, e.g., mandated by a safety standard. In our approach we utilise the safety contracts to capture the necessary information about a component from artefacts obtained out-of-context and show how argument-fragments can be generated for such components.

The structure of the paper is as follows: In Section 2 we present background information. In Section 3 we present the rationale behind our approach and how the generation of argument-fragments can be performed. In Section 4 we illustrate the approach for the Fuel Level Estimation System, and in Section 5 we provide a discussion of our approach. We present the related work in Section 6, and conclusions and future work in Section 7.

## 2 Background

In this section we introduce FLES that we use to illustrate our approach. We also provide some brief information on safety contracts based on our previous work; and Goal Structuring Notation, the argumentation notation we use for documenting safety case argument-fragments.

### 2.1 Illustrative Example: The Fuel Level Estimation System

In this subsection, based on [8], we provide brief but essential information related to FLES and the hazard analysis performed on it. We limit our attention to some bits of information that we use in illustrating the generation of argument-fragments.

FLES is based on a real estimation system used in Scania trucks with liquid fuel. The component-based architecture of FLES is shown in Fig. 1. The *Estimator* component estimates the volume of fuel in a vehicle's tank based on the sensor data obtained from the *Fuel Tank* and the *Engine Management System* (EMS). The received sensor values go through a series of transformations and filtering to handle any fluctuations in the sensed fuel level value. The estimated value is converted into percentage, passed to the *Presenter* and presented to the driver of the vehicle through the *Fuel Gauge* mounted on the dashboard. Due to dependencies of the transformations to the physical properties of sensors and its environment (e.g., size of the tank), these parameters are made configurable to make *Estimator* usable in different variants of the system.

The hazard analysis performed on the system reveals that if the fuel level displayed on the fuel gauge is higher than the actual fuel level in the tank then the vehicle could run out of fuel without the driver noticing, which would cause a sudden engine stop. If this happens while driving on e.g., a highway, the consequences could be catastrophic. Although there are other hazards in the system, this is the only hazard we use in illustrating our approach.

The safety analysis, as recommended by ISO 26262, starts by identifying at least one Safety Goal (SG) for each hazard, then for every safety goal, corresponding Functional Safety Requirements (FSRs) are derived and finally, Technical Safety Requirements (TSRs) are derived from the FSRs. We consider the following SGl and derived FSR:

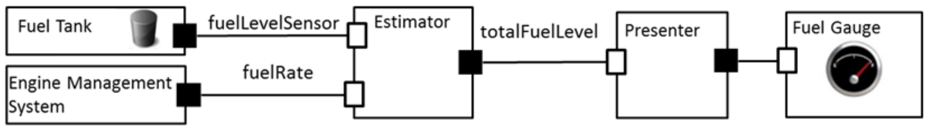

**Fig. 1.** Fuel Level Estimation System

- *SG1*: FLES shall not show higher fuel level on the fuel gauge than the actual fuel in the vehicle's tank;
- *FSR1*: Estimator shall not provide value of the estimated fuel level that deviates more than -5% from the actual fuel-level in the tank.

Additionally, the engine status signal provided by EMS should not be older than 0.3 seconds. An older value could result in a too high deviation from the actual fuel consumption that may cause deviation in the estimated fuel level value.

## 2.2 Strong and Weak Contracts

Our extension of the traditional contract-based formalism with strong and weak contracts allows for distinguishing between properties that are context-specific and properties that must hold for all contexts [14].

A traditional assumption/guarantee contract $C = \langle A, G \rangle$ is composed of assumptions $A$ and guarantees $G$, where a component offers the guarantees $G$ if its assumptions $A$ on its environment are satisfied [6]. As an illustrative and simplified example based on the system we presented in Section 2.1, we specify a contract for Estimator with assumptions that if both the fuel level and fuel rate are provided with sufficient accuracy, Estimator guarantees that the total estimated fuel level it provides will be with certain accuracy.

Strong contracts $\langle A, G \rangle$ are composed of strong assumptions (A) and strong guarantees (G), and weak contracts $\langle B, H \rangle$ of weak assumptions (B) and weak guarantees (H) [15]. While strong assumptions must hold in order for a component to be used in any context, weak assumptions and guarantees just provide additional information for particular contexts. We say that a component, described by a set of safety contracts, is compatible with a certain context if all of its strong assumptions are satisfied by the environment. The weak contracts ensure that in all compatible contexts where the weak assumptions (B) are satisfied, the component offers the weak guarantees (H). For example, strong contracts could assume input type, range, or minimum amount of stack required and guarantee similar properties. On the other hand, weak contracts assume configurable parameters such as tank or sensor parameters in FLES and guarantee different behaviour of the component dependant on those parameters such as different accuracy of the output or specific timing behaviour.

## 2.3 Goal Structuring Notation

In this paper, we use Goal Structuring Notation (GSN) [2] for expressing safety case argument-fragments. GSN is a graphical argumentation notation that can be used to specify elements of any argument. Some of the basic elements of GSN are illustrated in Fig. 2 and their semantics is given in the following list:

- *Goal*: a claim or a sub-claim that should be supported by the underlying argument. It can be broken down to several sub-goals (sub-claims).
- *Strategy*: describes a method used to develop a goal into additional sub-goals.
- *Context*: represents the domain/scope of the element it is connected to.
- *Solution*: describes the evidence that the connected goal has been achieved.
- *Undeveloped element*: states that the element to which the symbol is attached requires further development.
- *InContextOf*: used to connect context with goals.
- *SupportedBy*: used to show relationship of inference between goals in the argument, or to show that certain evidence is supporting a goal.
- *Away goal*: used to specify a module in which the goal is further developed.

For the sake of clarity it must be noted that the context element can be used to simply enrich or clarify the statements of the elements it is connected to. Besides the basic symbols, we additionally use a notational extension that supports abstract argument patterns [2]. More specifically, to denote a variable we use the curly brackets within statements; to denote generalised n-ary relationships between GSN elements we use the *supportedBy* relationship with a solid circle; to denote a choice, either 1-of-n or m-of-n selection, we use a solid diamond, which can be paired, using a simple connector line, with an *Obligation* element represented by an octagon symbol, stating condition for the choice selection.

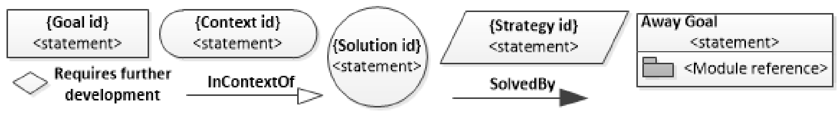

**Fig. 2.** Basic elements of the Goal Structuring Notation

## 3 Composable Arguments Generation

The aim of this section is twofold: (1) to explain the rationale underlying our approach to (semi)automatic generation of argument-fragments, and (2) to explain how the generation can be performed. The latter is done by

- providing a component meta-model, developed to capture the relationships between the safety contracts, safety requirements and evidence in an out-of-context setting, and being sufficient to provide us with the information required for argument-fragment generation,
- presenting a conceptual mapping of the meta-model elements to a subset of the basic GSN elements to provide better understanding of the transition from the meta-model to the argument-fragment,
- presenting an overview of the argument-fragment architecture, and by
- providing a set of rules for the argumentation-fragment generation.

## 3.1 Rationale of the approach

In our work we focus on safety-relevant components developed and prepared for safety certification independently of the system in which they will be used. To develop such components, the engineer must assume some safety requirements that might be required when the component is used in a context. To prepare components for certification, safety engineers need to capture safety-relevant properties of the component that show how the safety requirements allocated to the component are met. To do that, we use our notion of strong and weak contracts.

It is worth to point out that the safety requirements and the safety contracts we use are closely related, but not the same. The safety contracts contain information about the actual behaviour of the component. On the other hand, the safety requirements contain information about what a particular context/system requires from the component. While the safety requirements vary between contexts, the safety contracts should be correct regardless of the context. This is important to enable reuse of out-of-context components. As an illustration, consider FLES example requirement "Estimator shall send a valid value in totalFuelLevel within 2 seconds from when the Electronic Control Unit starts". This is a requirement on Estimator in this particular context and should not be specified within the Estimator's safety contract in that form. In the safety contract we should rather specify the actual time Estimator needs to send the totalFuelLevel. This makes the contracts independent of the context in which out-of-context component can be used, which allows us to use the knowledge captured within the contracts for all contexts in which the contracts are satisfied. The strong contracts denote properties that must be argued about in argument-fragments for every context, while the weak contracts will be argued about only if associated with a safety requirement within a particular context.

In order to guarantee the actual behaviour of the component, as specified in the safety contracts, we need to provide evidence about confidence in the contract. We categorise the evidence that supports the confidence in the contracts in terms of completeness, correctness and consistency, as follows: (1) completeness refers to whether contracts have captured all the needed properties of the component and the environment, (2) correctness refers to whether the contracts are correct with respect to associated requirements and (3) consistency refers to whether the contracts are not contradicting each other.

When using an out-of-context component in a particular context, a set of actual safety requirements (e.g., FSR or TSR) is allocated to the component. One of the roles of an argument-fragment is to show that these requirements are met. As safety contracts can be used to address different types of requirements, we are developing our approach without focusing on a particular class of requirements.

The (semi)automatic generation of argument-fragments from the safety contracts enables us to reduce the effort safety engineers need to dedicate for creating a set of argument-fragments. These fragments could be created for several contexts in which the component could be used. By speeding up both the integrator's and the developer's activities related to documenting a safety case, we enable them to focus on activities related to their knowledge about the system, by capturing this knowledge in the safety contracts.

## 3.2 Component meta-model

Our component meta-model in Fig. 3 is presented as an UML class diagram. This diagram captures the relationships between the assumed requirements, safety contracts and evidence, as described in Section 3.1. Our meta-model is based on the SafeCer component meta-model [7], which we have adapted, focusing only on its out-of-context part. Instead of associating argument-fragments (that may contain information not relevant for a specific context) with a component, we associate evidence and safety requirements directly with contracts to facilitate generation of context-specific argument-fragments.

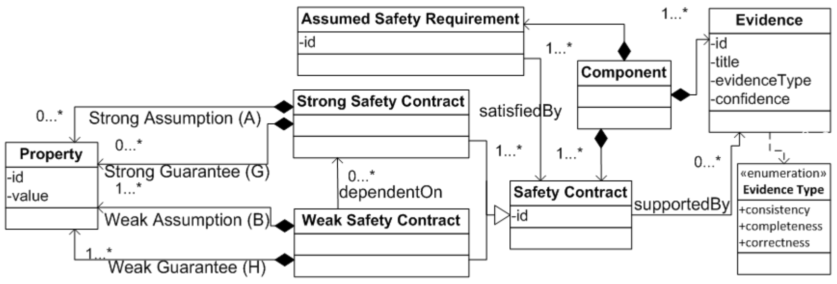

**Fig. 3.** Component and safety contract meta-model

The meta-model specifies a component that is composed of safety contracts, evidence and the assumed safety requirements. Each assumed safety requirement is satisfied by at least one safety contract, and each safety contract can have supporting evidence. Additionally, we assume that there is at least one evidence provided with the component supporting the consistency of the contracts. The safety contract elements in the meta-model are covering both the strong and weak safety contracts explained in Section 2.2. It should be noted that, based on the SafeCer component meta-model, the components can be composite i.e., a set of interconnected subcomponents, and can represent a (sub)system.

## 3.3 Conceptual mapping of the component meta-model to GSN

As mentioned in Section 2.3, GSN is used for documenting safety cases by expressing arguments and supporting evidence to show that the safety claims are satisfied. At the same time, as described in Section 3.2, our component meta-model captures the component safety claims in the safety contracts, supported by the associated evidence, with the goal to argue the satisfaction of the safety requirements. The conceptual mapping between the meta-model and GSN is depicted in Table 1.

**Table 1.** Conceptual mapping between the meta-model and GSN elements

| **The component meta-model elements** | **GSN-elements** |
|---|---|
| Properties representing guarantee(s)<br>Assumed safety requirement(s) | Goals |
| Evidence | Solutions |
| Properties representing assumption(s) | Contexts |

In order to build an argument structure from the safety contracts, we need to map the meta-model elements to the GSN elements. Our aim is to, based on our meta-model, develop an argument-fragment that addresses the following:

1. *Compatibility of a component with a context*: to show satisfaction of strong contracts of the component by the context, as described in Section 2.2. Besides satisfaction, confidence in contracts needs to be addressed using associated evidence.
2. *Satisfaction of safety requirements*: to show that a safety requirement is satisfied we need to argue both, that weak contracts related to the safety requirement are satisfied, and that the set of the related contracts is sufficient to show that the requirement is satisfied.
3. *Confidence in contracts*: showing only that a contract is satisfied by a context is not enough. Evidence about confidence in the contract should be provided also. We provide evidence about confidence in contracts in terms of completeness, correctness and consistency as described in Section 3.1.

The satisfaction of a contract, as described in Section 2.2, means that the contract guarantees are offered. Consequently, properties representing the safety contract guarantees in the meta-model as well as the assumed safety requirements correspond to goals in GSN. Furthermore, we use evidence from the meta-model related to consistency, correctness and completeness as solutions within GSN. To clarify the context of our goals, we make context statements providing properties representing the assumptions of the safety contracts.

## 3.4 Overview of the architecture of the resulting argument-fragment

Given the meta-model in Section 3.2, we propose to generate the resulting argument-fragment based on the mapping provided in Section 3.3.

In the argumentation-fragment generation we will follow a pattern that for a component, say $x$, with a top-level goal, say $G1$, in a series of successive steps will generate the corresponding argumentation fragment. We start by decomposing the goal $G1$ into three sub-goals, as shown in Fig. 4. We first argue satisfaction of all the strong contracts of $x$ in the goal $G2$. Then, we provide evidence for the consistency of all the contracts associated with $x$ in the goal $G4$ and finally, we argue over satisfaction of the requirements by the related contracts in the goal $G3$. We now further develop the goal $G3$ and leave the goals $G2$ and $G4$ undeveloped, as they will be explored later.

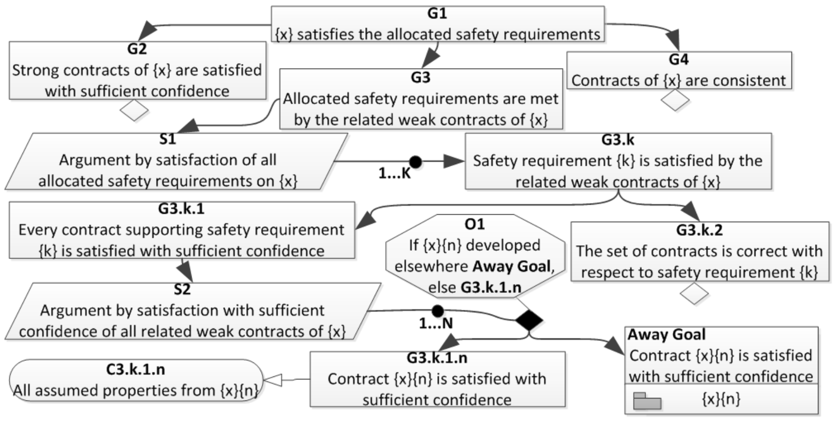
G1
{x} satisfies the allocated safety requirements
G2
Strong contracts of {x} are satisfied with sufficient confidence
G3
Allocated safety requirements are met by the related weak contracts of {x}
G4
Contracts of {x} are consistent
S1
Argument by satisfaction of all allocated safety requirements on {x}
1...K
G3.k
Safety requirement {k} is satisfied by the related weak contracts of {x}
G3.k.1
Every contract supporting safety requirement {k} is satisfied with sufficient confidence
O1
If {x}{n} developed elsewhere Away Goal, else G3.k.1.n
G3.k.2
The set of contracts is correct with respect to safety requirement {k}
S2
Argument by satisfaction with sufficient confidence of all related weak contracts of {x}
1...N
Away Goal
Contract {x}{n} is satisfied with sufficient confidence
{x}{n}
C3.k.1.n
All assumed properties from {x}{n}
G3.k.1.n
Contract {x}{n} is satisfied with sufficient confidence

**Fig. 4.** Safety requirements satisfaction goal sub-structure

We further develop the goal $G3$ by applying the strategy $S1$ to argue over satisfaction of all safety requirement allocated to component $x$. For every safety requirement $k \in [1, K]$ where $K$ is the number of allocated requirements, a goal $G3.k$ is created, stating satisfaction of the requirement by the related contracts. We further break down the $G3.k$ goal into two sub-goals: (1) $G3.k.1$ arguing over satisfaction of every supporting contract of the requirement k, and (2) $G3.k.2$ providing associated evidence that the related safety contracts supporting the safety requirement $k$ are correct with respect to the requirement. We first focus on the $G3.k.1$ goal, and leave the $G3.k.2$ goal undeveloped, as it will be explored together with other parts of the argument referring to evidence.

When arguing over satisfaction with sufficient confidence of a set of contracts, we use the same strategy whether we argue over all the strong contracts ($G2$) or the weak contracts that support the safety requirements. To further develop the $G3.k.1$ goal, we apply the strategy $S2$ to argue over satisfaction with sufficient confidence over every related contract and reach the choice represented by obligation $O1$. If a goal has been developed elsewhere to support a contract $n$ we create an away goal, otherwise we create a goal $G3.k.1.n$ for every contract $n$ arguing over its satisfaction, where $n \in [1, N]$, with $N$ being the number of related contracts to the requirement $k$. In order to further clarify the goal $G3.k.1.n$ we provide assumed properties of the contract $n$ as a goal context.

As shown in Fig. 5, to argue that a safety contract $n$ is satisfied with sufficient confidence we break down the goal $G3.k.1.n$ into two sub-goals: (1) $G3.k.1.n.1$ arguing over satisfaction of every safety contract $m$ that supports the assumed properties of the contract $n$, where $m \in [1, M]$ and $M$ is the number of contracts supporting the contract $n$, and (2) $G3.k.1.n.2$ providing attached evidence about the completeness of contract $n$. We further develop the goal $G3.k.1.n.1$ by applying the strategy $S3$ to argue over satisfaction of every supporting contract $m$ and create a sub-goal $G5.m$ arguing that the corresponding assumed property

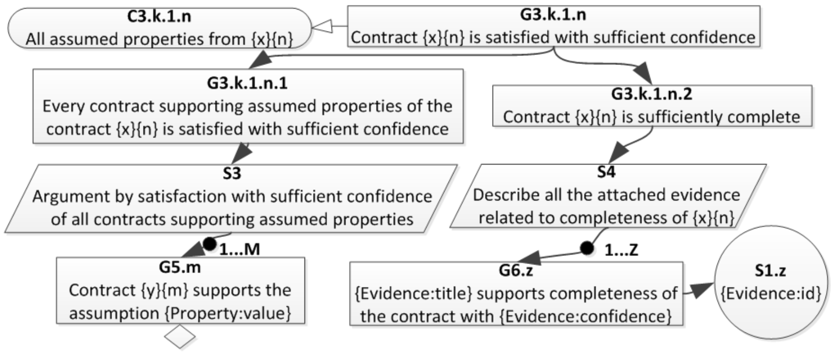
C3.k.1.n
All assumed properties from {x}{n}
G3.k.1.n
Contract {x}{n} is satisfied with sufficient confidence
G3.k.1.n.1
Every contract supporting assumed properties of the contract {x}{n} is satisfied with sufficient confidence
G3.k.1.n.2
Contract {x}{n} is sufficiently complete
S3
Argument by satisfaction with sufficient confidence of all contracts supporting assumed properties
S4
Describe all the attached evidence related to completeness of {x}{n}
1...M
1...Z
G5.m
Contract {y}{m} supports the assumption {Property:value}
G6.z
{Evidence:title} supports completeness of the contract with {Evidence:confidence}
S1.z
{Evidence:id}

**Fig. 5.** Contract satisfaction with confidence goal sub-structure

of the contract $n$ is satisfied by the supporting contract $m$. To develop the goal $G5.m$ we apply the same strategy as for the goal $G3.k.1$.

For developing the three arguments that present the attached evidence related to completeness, correctness and consistency, represented by the goals $G3.k.1.n.2$, $G3.k.2$ and $G4$, we develop the argument inspired by the "Specification Argument Pattern" [4]. Unlike in that work, we define the three types of evidence differently, as described in Section 3.1. The goal $G3.k.1.n.2$ is developed by applying a strategy $S4$ to argue over every attached evidence of the specific type. For every evidence a goal is created claiming with what level of confidence does this goal support the completeness/consistency/correctness and the evidence reference is provided as the solution to the goal.

## 3.5 Rules for generation of component argument-fragments

Given the argument structure in Section 3.4 and the component meta-model we can define a sequence of transformation rules that facilitate (semi)automatic generation of argument-fragments. Our goal is not only to transfer all the information provided by the safety contracts into the argument-fragment, but also to point out the goals that need further development and thus alert safety managers. For this we use undeveloped goals within the argument-fragments. We provide the rules similarly as in [9]. We create an argument-fragment for a component $x$ by using the following rules:

R1. Create the top-level goal $G1$: "$\{x\}$ satisfies the allocated safety requirements". Develop the goal $G1$ further by creating three sub-goals:
   (a) $G2$: "Strong contracts of $\{x\}$ are satisfied with sufficient confidence".
   (b) $G3$: "Allocated safety requirements are met by the related weak contracts of $\{x\}$".
   (c) $G4$: "Contracts of {x} are consistent".

R2. Further develop the goal $G3$ and for every allocated safety requirement $k$ create a goal $G3.k$ "Safety requirement $\{k\}$ is satisfied by the related weak contracts of $\{x\}$" and develop this goal further by creating two sub-goals:

(a) $G3.k.1$: "Every contract supporting safety requirement $\{k\}$ is satisfied with sufficient confidence".
(b) $G3.k.2$: "The set of contracts is correct with respect to safety requirement $\{k\}$".

R3. Further develop the goal $G3.k.1$ by developing an argument for every safety contract $n$ of the component $x$, associated with the safety requirement $k$. If the contract satisfaction module is developed elsewhere in the argument provide an away goal, otherwise create a sub-goal $G3.k.1.n$ "Contract $\{x\}\{n\}$ is satisfied with sufficient confidence" and provide properties representing the assumptions of the contract $\{n\}$ as the goal context $C3.k.1.n$. Further develop the sub-goal:
(a) $G3.k.1.n.1$: "Every contract supporting assumed properties of the contract $\{x\}\{n\}$ is satisfied with sufficient confidence". For every contract $m$ supporting the assumed property $p$ of the contract $n$ create a sub-goal $G5.m$: "Contract $\{y\}\{m\}$ supports the assumption $\{p\}$ ", where $m$ is specified for a component in environment of $x$, say $y$.
(b) $G3.k.1.n.2$: "Contract $\{x\}\{n\}$ is sufficiently complete".

R4. The goal $G5.m$ is developed further in the same way as $G3.k.1$ and the goal $G2$ is developed further in the same way as the goal $G3.k.1.n$.

R5. Goals $G3.k.1.n.2$, $G3.k.2$ and $G4$ are developed further in the same way for the list of attached evidence of the corresponding type, respectively, completeness, correctness and consistency. For every evidence $z$ from the corresponding list of evidence type:
(a) Create a goal $G6.z$: "$\{Evidence : title\}$ supports $\{EvidenceType\}$ of the contract with $\{Evidence : confidence\}$".
(b) Attach a solution $S1.z$ to the goal $G6.z$ with $Evidence : id$ as reference.

R6. If no evidence of a particular type is provided, an undeveloped goal is used to indicate that the goal should be further developed.

It should be noted that, based on Rule $R4$, we can generate argument-fragments for a composite component by iterating through hierarchical structure. Applying the rules to an out-of-context component will generate an incomplete argument-fragment since not all relevant claims can be captured out-of-context. Such claims are left undeveloped, e.g., correctness of contracts with respect to a safety requirement. Hence further development of the argument-fragment is required to address all the undeveloped claims.

## 4 Argument-fragment for FLES

In this section we provide safety contracts for the Estimator and EMS components of FLES, as well as show the generation of an argument-fragment for Estimator.

**Table 2.** Safety contracts for the Estimator component

| |
|---|
| **A1**: fuelLevelSensor within [0,5] AND fuelRate within [-1,3212]<br>**G1**: totalFuelLevel within $[-1, 100]$<br>$\mathbf{E}_{A1,G1}$: Sw architecture design specification, Sw architecture verification report |
| **B1**: (fuelLevelSensor within correct range AND fuelLevelSensor does not deviate more than 10% from the actual fuel level value AND fuelLevelSensorParameter=10) OR (fuelRate within [0,3212] AND fuelRate does not deviate more than 1% from the actual engine consumption value AND Tank size within [230-1000])<br>**H1**: totalFuelLevel does not deviate more than -1% from the actual fuel level value<br>$\mathbf{E}_{B1,H1}$: Simulation of the Estimator component under assumed conditions |

## 4.1 The safety contracts

The strong and weak contracts for Estimator addressing the requirement $FSR1$ of FLES are shown in Table 2. The strong contract assumes the allowed ranges of inputs and guarantees the possible outputs of the component. The evidence supporting the completeness of strong contract $\langle A1, G1\rangle$ includes the software architecture design specification and the corresponding verification report.

As described in Section 2.1, the quality of the totalFuelLevel output of the Estimator component is dependent on relevant parameters and the quality of inputs. The weak contract $\langle B1, H1\rangle$ of Estimator guarantees that the deviation of the totalFuelLevel from the actual fuel level is less than or equal to -1% if assumptions on either fuelLevelSensor and parameters related to it, or fuelRate and parameters related to it, are satisfied. The corresponding evidence is obtained by simulation of Estimator under the assumed conditions, and the simulation report is attached as evidence supporting the contract completeness.

The EMS component safety contracts related to the Estimator component are provided in Table 3. The EMS strong contract is similar to the one for the Estimator component, ensuring the input and output port ranges. The weak contract $\langle B2, H2\rangle$ guarantees that the deviation of the estimated fuel consumption does not exceed 0.4% of the actual fuel consumption under the assumed engine parameters and freshness of the information obtained from the engine. A simulation of the EMS component's behaviour under the stated conditions is attached as an evidence to support contract completeness.

**Table 3.** Safety contracts for the EMS component

| |
|---|
| **A2**: engineStatus within [a,b]<br>**G2**: fuelRate within [-1,3212]<br>$\mathbf{E}_{A2,G2}$: Sw architecture design specification, Sw architecture verification report |
| **B2**: Engine parameters=20 AND engineStatus delay under 0.3 seconds<br>**H2**: fuelRate does not deviate more than 0.4% from the actual fuel consumption<br>$\mathbf{E}_{B2,H2}$: Simulation of the fuel consumption estimation under assumed conditions |

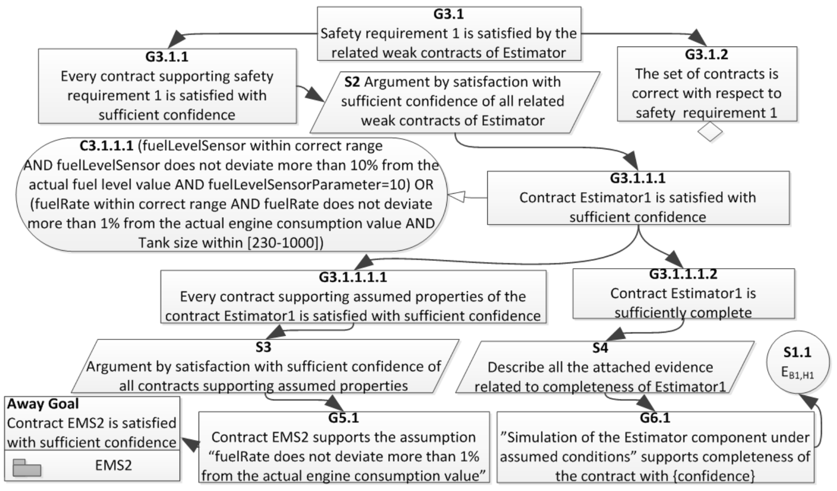
G3.1 Safety requirement 1 is satisfied by the related weak contracts of Estimator
G3.1.1 Every contract supporting safety requirement 1 is satisfied with sufficient confidence
S2 Argument by satisfaction with sufficient confidence of all related weak contracts of Estimator
G3.1.2 The set of contracts is correct with respect to safety requirement 1
C3.1.1.1 (fuelLevelSensor within correct range AND fuelLevelSensor does not deviate more than 10% from the actual fuel level value AND fuelLevelSensorParameter=10) OR (fuelRate within correct range AND fuelRate does not deviate more than 1% from the actual engine consumption value AND Tank size within [230-1000])
G3.1.1.1 Contract Estimator1 is satisfied with sufficient confidence
G3.1.1.1.1 Every contract supporting assumed properties of the contract Estimator1 is satisfied with sufficient confidence
G3.1.1.1.2 Contract Estimator1 is sufficiently complete
S3 Argument by satisfaction with sufficient confidence of all contracts supporting assumed properties
S4 Describe all the attached evidence related to completeness of Estimator1
S1.1 $E_{B1,H1}$
Away Goal Contract EMS2 is satisfied with sufficient confidence
EMS2
G5.1 Contract EMS2 supports the assumption "fuelRate does not deviate more than 1% from the actual engine consumption value"
G6.1 "Simulation of the Estimator component under assumed conditions" supports completeness of the contract with {confidence}

**Fig. 6.** A part of the resulting argument-fragment

## 4.2 The resulting argument-fragment for the Estimator component

In Fig. 6 we provide a part of the argument-fragment for $FSR1$ of FLES, allocated to the Estimator component and associated with the Estimator contract $\langle B1, H1\rangle$ denoted as $Estimator1$ within the argument. By using the rules from Section 3.5, we generate an argument-fragment from the provided safety contracts to argue over satisfaction of $FSR1$ by showing that the requirement is satisfied by the related $Estimator1$ contract. The argument for satisfaction of $Estimator1$ contract is developed to show the associated evidence supporting its completeness, and point to the away goals supporting its assumed properties. Due to space limitations we show only an away goal supporting $Estimator1$ assumed property related to the fuelRate deviation and supported by the EMS $\langle B2, H2\rangle$ contract, denoted as $EMS2$ within the argument. The generated argument-fragment contains some properties that could be captured in an out-of-context setting and should be further developed to cover all relevant properties not captured within the contracts.

## 5 Discussion

As seen in the example in Section 4 we are able to generate a partial argument-fragment based on the component meta-model in Section 3.2. We support the confidence in contract completeness by associating the supporting evidence with the contracts. At the same time, by making the contracts related to the actual behaviour of the component and not to particular safety requirements, we are able to use the contracts to address different context-specific safety requirements.

The presented approach allows us to use the safety claims captured for an out-of-context component to develop context-specific argument-fragments. The resulting argument-fragment for a particular context should not include information relevant for all contexts, but only the information relevant for the particular context. By automating the generation of argument-fragments from safety contracts we speed up the creation of such argument-fragments for different contexts. The argument presented in Section 4 does not present all the aspects an argument should cover, such as failure modes or process-based arguments, but it provides an illustration of how the contracts can be used to generate argument-fragments. Contracts can be used to capture different safety aspects of components, e.g., failure behaviour. The resulting argument quality depends on the quality and variety (e.g., in terms of aspects) of the provided contracts.

The amount of work that still needs to be performed for a specific system depends on the abstraction level at which we allocate the safety requirements to components that have their safety contracts specified. If we connect the requirements with the contracts at higher levels of abstraction, based on the compositional nature of our approach a more complete argument-fragment could be generated. According to ISO 26262, SEooC cannot be an item, i.e., a system implementing a complete functionality, but it can be a subsystem or a subcomponent of an item. Hence we focused on lower level components and how to reduce efforts needed to generate their argument-fragments.

The problem of automation and reuse of safety analyses and safety reasoning within the safety cases is a sensitive issue, especially since safety is a system property and needs to be reasoned about for the particular system. As mentioned in [13], the goal of automation is not to replace human reasoning, but to focus it on areas where they are best used. Similarly, in this work we are not aiming at eliminating human reasoning from the process of safety reasoning and argumentation, but to support it by providing automation of more clerical tasks.

## 6 Related Work

Generating safety case arguments to increase efficiency of safety certification has been a topic of many recent works. While some consider different notions of assumption/guarantee contracts for that purpose [17, 10] others directly build upon safety requirements [9, 3].

Assume/guarantee contracts are used in [17] to capture the vertical dependencies between a software application and a hardware platform that enables automatic generation of application specific arguments. The work presents a model-based language for specifying demanded and guaranteed requirements between the applications and platforms. The language allows for capturing restricted set of properties, whereas the contract formalism we base our work on is more expressive and offers support for easier out-of-context to in-context reuse of components. Also, [17] does not provide means for generating arguments from the captured contracts.

An approach where "informal" contracts are used for safety-case generation is proposed in [10]. The approach uses Dependency-Guarantee Relationships (DGRs) that correspond to our contracts. It derives an argument for a module by using all the DGRs of the module to build an argument relying on dependencies from other modules. In contrast to this approach, we take in consideration different types of evidence that need to be provided with the safety contracts and components, including compatibility of a component with a particular context.

A method for automated generation of safety case arguments based on an automatic extraction of information from existing work-products is presented in [3]. The generated argumentation consists of summaries of different work-products created within a project. Similarly, a methodology for safety case assembly from artefacts required to satisfy some process objectives is presented in [9]. The work provides a set of transformation rules from captured safety requirements to safety case arguments. While these methods are useful for generating a safety case argument from a set of safety requirements that are related to existing work-products, they do not as we do consider reuse of out-of-context components developed and prepared for certification.

## 7 Conclusion and Future Work

In this paper we have presented an approach for generating safety case argument-fragments from safety contracts for out-of-context components developed and prepared for safety certification independently of the system in which they will be used. The approach allows us to speed up the creation of context-specific argument-fragments. More specifically, we have presented an overview of the argument-fragment architecture and provided a set of rules for generating the argument-fragments from the safety contracts, including illustrating the application of the rules with an example. We can conclude that safety contracts provide a good basis for generating argument-fragments and in that way allow safety engineers to focus more on capturing the knowledge about the system rather than spending time on documenting a safety case.

In our future work, we plan to refine our component meta-model, e.g., to provide support for different classes of requirements. Consequently, this refinement entails co-evolution of the generation rules. We also plan to implement the provided rules within an existing tool that supports a contract formalism, e.g., the CHESS-toolset [1]. To show the scalability of our approach we aim at using it for more complex case studies, e.g., for a larger number of safety requirements. Further more, we plan to explore how our approach could be used to reduce some of the common argument fallacies [11] related to the structure of arguments. Moreover, it is worthwhile investigating usage of our approach for safety case maintenance and change management.

## Acknowledgements

Thanks to Iain Bate for useful discussions and comments. This work is supported by the Swedish Foundation for Strategic Research (SSF) via project Synopsis as well as EU and Vinnova via the Artemis JTI project SafeCer.

## References

[1] CHESS-toolset: http://www.chess-project.org/page/download.

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

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

[4] I. Bate and P. Conmy. Assuring Safety for Component Based Software Engineering. In *15th IEEE International Symposium on High Assurance Systems Engineering (HASE)*, January 2014.

[5] I. Bate, H. Hansson, and S. Punnekkat. Better, faster, cheaper, and safer too - is this really possible? In *17th IEEE Int'l Conf. on Emerging Technologies for Factory Automation (ETFA)*. IEEE, September 2012.

[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] J. Carlson et al. "Generic component meta-mode, Version 1.0" SafeCer, Deliverable D132, November 2013.

[8] R. Dardar. Building a Safety Case in Compliance with ISO 26262 for Fuel Level Estimation and Display System. Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden, 2014.

[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, September 2012.

[10] J. L. Fenn, R. D. Hawkins, P. Williams, T. P. Kelly, M. G. Banner, and Y. Oakshott. The who, where, how, why and when of modular and incremental certification. In *2nd International Conference on System Safety (ICSS)*. IET, 2007.

[11] W. S. Greenwell, J. C. Knight, C. M. Holloway, and J. J. Pease. A taxonomy of fallacies in system safety arguments. In *24th International System Safety Conference (ISSC)*, 2006.

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

[13] J. Rushby. Logic and epistemology in safety cases. In *32nd International Conference on Computer Safety, Reliability, and Security (SafeComp)*. Springer-Verlag, September 2013.

[14] 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.

[15] 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, November 2013.

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

[17] B. Zimmer, S. Bürklen, M. Knoop, J. Höfflinger, and M. Trapp. Vertical safety interfaces–improving the efficiency of modular certification. In *30th International Conference on Computer Safety, Reliability and Security (SafeComp)*. Springer-Verlag, September 2011.