Contract number: ITEA2 – 10039

Safe Automotive soFtware architEcture (SAFE)

ITEA Roadmap application domains:

Major: Services, Systems & Software Creation

Minor: Society

ITEA Roadmap technology categories:

Major: Systems Engineering & Software Engineering

Minor 1: Engineering Process Support

WP3

Deliverable D3.1.3: Proposal for extension of Meta-model for safety case modeling and documentation

Due date of deliverable: 31/03/2013

Actual submission date: 28/03/2013

Start date of the project: 01/07/2011 **Duration:** 36 months

Project coordinator name: Stefan Voget

Organization name of lead contractor for this deliverable: fortiss

Editor: Maged Khalil (fortiss GmbH)

Contributors: Maged Khalil (fortiss GmbH), Eduard Metzker (Vector Informatik GmbH)

Reviewers: Eduard Metzker (Vector Informatik GmbH)

Revision chart and history log

Version	Date	Reason
0.1	2012-09-19	Initialization of document
0.2	2012-10-18	Input to Introduction Section 4
0.3	2012-11-07	Input to Overview on ISO Section 5
0.4	2012-11-16	Input to EAST-ADL Overview Section 7
0.5	2012-11-16	Input to Overview on ISO Section 5
0.6	2012-11-23	Input to Methodology Section 6
0.7	2012-11-29	Further Input to Methodology Section 6
0.8	2012-12-20	Editing and Input in various sections
0.9	2013-02-13	Input to SAFE Meta-model Contribution Section 8
0.10	2013-02-15	Introduction of Patterns Section 6.3
0.11	2013-03-08	Editing and Input in various sections
0.12	2013-03-24	First version ready for review
0.13	2013-03-26	Incorporation of first review comments
0.14	2013-03-27	Updated version reviewed. Input to Section 8.3.
1.0	2013-03-28	Incorporation of review comments, finalization of deliverable

1 Table of contents

2 List of figures

3 Executive Summary

The work task 3.1.3 targets the topics of safety case modeling and documentation according to ISO 26262.

Besides giving an overview on the relevant sections of ISO 26262 the requirements allocated to WT 3.1.3 which are resulting from the ISO 26262 analysis of WT 2.1 are presented. In an additional section, the current achievements on the requirements allocated to WT 3.1.3 are presented.

In addition to the previous mentioned overview the methodology for safety case documentation in accordance with ISO 26262 is presented. Since it is the objective to develop a Meta-model extension for safety cases the current version of EAST-ADL is analyzed. Moreover, the contribution of WT 3.1.3 to the SAFE Meta-model, which is based on EAST-ADL, is presented. Methods of how to use the provided safety case capability in a generic as well as in a pattern-based approach are explained.

4 Introduction and overview of document

The document at hand provides information about a methodology for safety case modeling and documentation and a proposal for the extension of the SAFE Meta-model that enables its use in a model-driven development environment. In the following subsection the scope of the work task as well as the structure of the document is presented.

4.1 Scope of WT 3.1.3

Embedded in work package 3, work task 3.1.3 deals with the safety case modeling including the ability to describe artifacts of the SAFE Meta-model in a safety case relevant context, as well as an expression of relations between artifacts in that context. The basis for this work task is the structured argumentation notation known as the Goal Structuring Notation (GSN) which is presented in section 6. WT 3.1.3 aims to provide a methodology for argumentation about safety cases and a Meta-model extension suitable for the mentioned topics to WT 3.5. Furthermore WT 3.1.3 explores how the provided methodology can be used in a pattern-based approach. In order to be able to do so, mainly the following artifacts and their interrelations are considered:

Hazard

Hazards represent the potential source of harm and form a key aspect of the hazard analysis and risk assessment and a focal point for safety activities. WT 3.1.1 provides a concept to express hazards in formal as well as informal formulation.

Hazardous Event

Hazardous events are relevant combinations of hazards and operational situations in a given operating mode. WT 3.1.1 develops a suitable representation including a concept for hazardous events shall enable the classification according to the parameters severity, probability of exposure, and controllability. Based on these parameters the ASIL classification is performed which is be supported by the Meta-model concept. Hazardous events can be categorized into general problem patterns that, in their turn, result in general solution patterns which can be collected into libraries that include the corresponding safety case template.

Safety Goal

WT 3.1.1 derives safety goals from hazardous events and enables the expressions and documentation of safety goals with their respective parameters and association with a SAFE state ("operating mode of an item without an unreasonable level of risk" [1]). The safety goal is the starting point for any safety case.

Functional Safety Concept

The functional safety concept details the approach that will be used to counteract or mitigate the effects of the hazardous event and satisfy the corresponding safety goal(s). Functional safety concept can be categorized into known solution patterns, e.g., use redundancy to overcome the low reliability of a single channel or a source of common cause failures.

Technical Safety Concept

The technical safety concept describes the actual implementation details of the corresponding functional safety concept. E.g., redundancy can be achieved by using a homogenous hardware redundancy pattern, or a software logical redundancy pattern, according to the requirements of the situation.

4.2 Structure of document

The document is structured as follows:

Subsequent to this introduction an overview on the parts of ISO 26262, which are relevant for safety case modeling and documentation, is given in section 5.

Within chapter 6, the methodology for safety case description is explained. To do this, in a first step some background information and motivation is given, followed by a general introduction to the methodology used. Safety case description elements and their relations are discussed in Section (6.2). Section (6.3) gives an overview of pattern-based approaches and details the use of solution patterns for the generation of safety case skeletons or templates. Finally, an overview of testing of safety cases is given in Section (6.4).

Section 7 deals with EAST-ADL. EAST-ADL is an architecture description language (ADL), which has been developed in various projects in which both automotive vendors and users cooperated, where the objective was to define an ADL tailored to the needs of the automotive industry. The current version published on their website (www.east-adl.info) is EAST-ADL V2.1 [44].

On the one hand, the current version of EAST-ADL and in particular the dependability part is described und studied from a safety case perspective. On the other hand, some proposed extensions from parallel research and industrial projects to this current version are explained which introduce or enhance the possibility to perform safety case documentation in compliance with ISO 26262.

The contribution of WT 3.1.3 to the SAFE Meta-model is described in section 8. Within this section an overview on the part of the Meta-model as well as a detailed description of the classes and links used to construct the Meta-model is presented. Moreover, an example for the application of the Meta-model for safety case documentation is presented.

In section 9 the relevant interdependencies to other work tasks of SAFE are depicted and explained.

Finally, in section 10 a summary and conclusion are given.

5 Overview on ISO 26262

Addressing the development process of electric / electronic components for passenger cars, the ISO 26262 "Road vehicles – Functional safety" came into effect in November 2011. This standard introduces a safety lifecycle which "encompasses the principal safety activities during the concept phase, product development, production, operation, service and decommissioning" ([1], part 2, p.3) and which can be seen as a guideline that demands a risk-assessment based development approach with seamless traceability.

Within this section, an overview on the relevant parts of ISO 26262 with regard to safety case modeling is given. The selection of the presented parts is based on the source of the SAFE requirements elicited in WT 2.1 which are allocated to WT 3.1.3.

However, as the purpose of a safety case is, simply put, to link safety goals to the solution fulfilling them using clear lines of argumentation, it is clear that safety case modeling capability is relevant for virtually all aspects of the ISO26262 activities covered by the SAFE Meta-model, in order to provide the link between generated artifacts and the requirements driving them.

In figure (1) an overview on the different parts of ISO 26262 is given, with relevant parts directly derived from WT2.1 requirements colored red and other safety case modeling relevant parts colored blue.

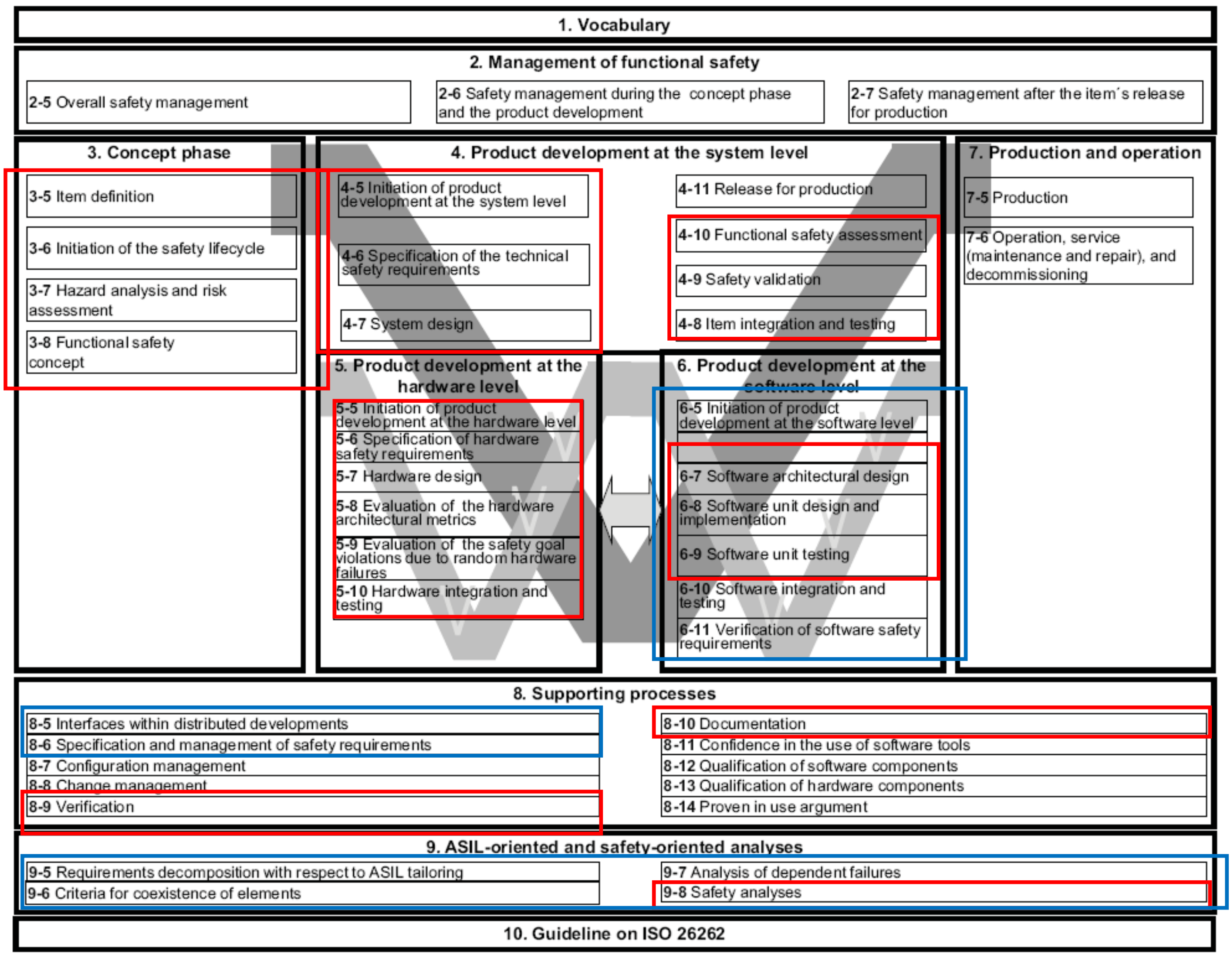
1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during the concept phase and the product development
2-7 Safety management after the item´s release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development at the system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
5. Product development at the hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Evaluation of the hardware architectural metrics
5-9 Evaluation of the safety goal violations due to random hardware failures
5-10 Hardware integration and testing
6. Product development at the software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Confidence in the use of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262

Figure 1: Overview on ISO 26262 (Relevant parts highlighted)

In the following, an overview on the relevant aspects from the respective ISO26262 parts is given.

Part 3: Concept Phase

The concept phase comprises mainly four different parts, namely the item definition, the initiation of the safety lifecycle, the hazard analysis and risk assessment, and the functional safety concept. These parts are explained in the following subsections.

Item Definition

The objective of the definition is to provide an overview on the item, the implemented functionalities and the dependencies as well as interactions of the item with the environment or other items of the vehicle. This information shall be provided in form of functional and non-functional requirements of the item. Moreover, the item definition includes a boundary description of the item as well as of elements of the item, i.e. a description of the interfaces and the expected as well as provided functionalities and interactions.

Initiation of the Safety Lifecycle

During the sub-phase of the initiation of the safety lifecycle it is distinguished between new developments and modifications of existing items. Depending on this the entire safety lifecycle or a tailored version needs to be applied.

Hazard Analysis and Risk Assessment

In general, the hazard analysis and risk assessment takes place based on the item definition and evaluates present risks without taking into account internal safety mechanisms of the item. This serves as the basis for defining safety goals and deriving the functional safety concept and its requirements, as shown in figure (2).

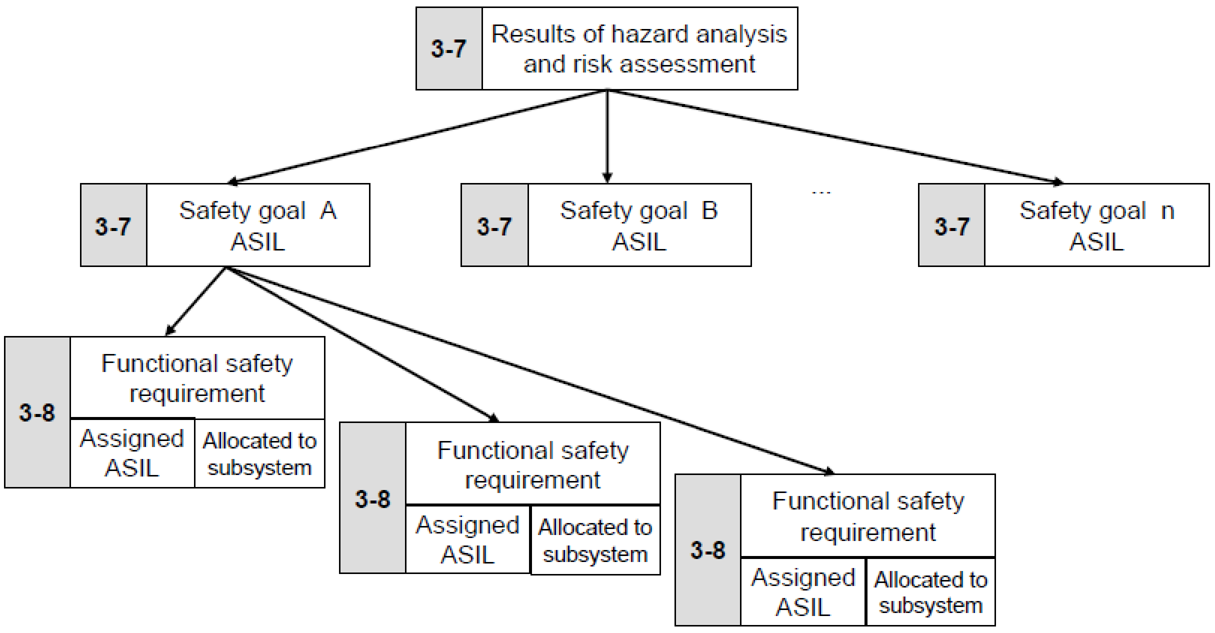

Figure 2: Concept phase activities after HA/RA carried out

In a first step of the analysis, possible operational situations that are scenarios which might occur during the vehicles lifetime are collected. In this step it is important also to cover situations that arise through foreseeable misuse of the vehicle. Subsequent to the definition of operational situations hazards which are related to the item need to be determined. Although the hazards need to be related to the item and are associated with a malfunction of the item, the description takes place on vehicle level, i.e. the resulting behavior at vehicle level needs to be determined. After identifying the hazards, relevant combinations of both, hazards and operational situations, are captured as hazardous events. These hazardous events are subject to classification according to the three parameters controllability, probability of exposure and severity. Based on the parameters the ASIL (Automotive Safety Integrity Level) is determined and assigned to the hazardous event. In case the determination of the ASIL leads to ASIL A, B, C or D, a safety goal has to be derived from the particular hazardous event. These safety goals are the top-level safety requirements for the item and serve as a basis for the later development of the functional safety concept.

Functional Safety Concept

Subsequent to the hazard analysis and risk assessment the functional safety concept is developed. The functional safety concept consists of functional safety requirements and preliminary architectural assumptions. The functional safety requirements which are derived from the safety goals are allocated to the elements of the item.

Part 4: Product Development – System Level

During this phase the development of the item from the system level perspective takes place. The process is based on the concept of a V-model. Starting point (on the upper left side) is the specification of the technical safety requirements which is followed by the development of the system architecture and the system design. The way up to the upper right point of the V-model is built by the integration, verification, validation and functional safety assessment activities.

Part 5: Product Development – Hardware Level

During this phase the development of the item from the hardware perspective is performed. The process is again based on a V-model, going down with the specification of hardware safety requirements as well as hardware design and implementation and back upwards with hardware integration and testing.

Part 6: Product Development – Software Level

During this phase the development of the item from the Software perspective is performed. The process is again based on a V-model, going down with the specification of Software safety requirements as well as Software design and implementation and back upwards with Software integration, testing and validation.

Part 8: Supporting Processes

The relevant requirements for WT 3.1.3 arise from two sections of part 8 (supporting processes), namely “Verification” and “Documentation”, with the following overview limited to these sections.

Verification

Within the section “Verification” requirements are given which need to be fulfilled in order to ensure that the work products comply with their requirements.

Documentation

Within the section “Documentation” requirements are given which need to be fulfilled in order to ensure that the work products and processes and all relevant links are properly documented.

<u>Part 9: Automotive Safety Integrity Level (ASIL)-oriented and Safety-oriented Analyses</u>

The assigned and accepted requirements for WT 3.1.3 arise from one section of part 9 (automotive safety integrity level (ASIL)-oriented and safety-oriented analyses), namely "Safety analyses". An over of only this section of part 9 is given.

Safety Analyses

With the help of the safety analyses consequences of faults and failures on functions, behavior and design of items and elements shall be examined. Moreover, the analyses provide information on causes and conditions that could lead to the violations of a safety goal or safety requirement. Additionally, the analyses contribute to the identification of new hazards not discovered during the hazard analysis and risk assessment.

6 Methodology for Safety Case Description

After presenting the relevant parts of ISO 26262 covered by the requirements from WP 2 allocated to WT 3.1.3, we will present the methodology we propose to express safety cases in this section.

6.1 Introduction

Although ISO 26262 requires looking at the risk emanating from the item without considering other elements of the vehicle architecture and without considering internal safety measures (cp. ISO 26262:3-2011, requirement 7.4.1.2), this risk itself is determined by the role of the item in the vehicle architecture. An example for this is that an EPS (electric power steering) system can be realized in a way that it can be overruled in any case by the driver. This would lead to a totally different classification compared to the realization of an EPS which cannot be overruled by the driver due to a too strong impact [45].

Therefore the model-based development process foreseen by SAFE has to take into account not only the item features but also all other elements / attributes that potentially contribute to the risk on vehicle level. The architecture suitable for the consideration of these needs has to fulfill the following aspects:

- there is a hierarchical architecture
- environmental aspects have to be distinguished
- functional / technical aspects have to be distinguished
- within technical aspects the hardware and software aspects need to be distinguished

The resulting architecture which is used in SAFE/SAFE-E [45] is presented in the following figure.

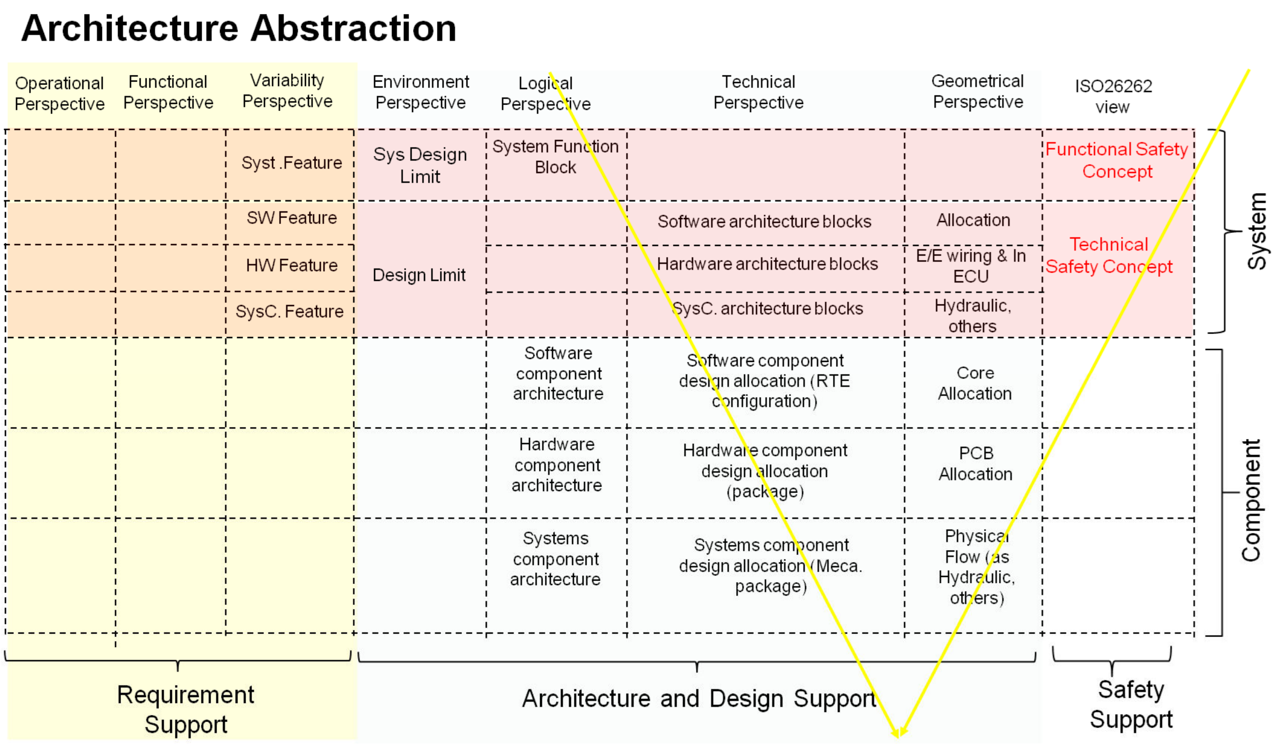

Figure 3: Overview on Structure of Architecture

Due to the structure of the architecture matrix shown the ASIL allocation could be different. Moreover, ASIL decomposition could be applied on any horizontal level, which has influences to the lower horizontal levels. Analogous to this, safety requirements could be allocated to different elements within the horizontal level; this implies that a safety mechanism could be implemented into a sensor or alternatively into the controller or the actuator. By applying graceful degradation also the technical behavior in case of failure could be different and again this would lead to different inheriting of safety requirements to lower horizontal level.

As already depicted in the previous section 5, ISO26262 requires a detailed traceability linking requirements and their fulfillment to the underlying architecture in a comprehensible manner. This link between safety goals and the solutions fulfilling them using clear lines of argumentation is, simply put, the purpose of a safety case.

A common definition of a safety case is “a documented body of evidence that provides a convincing and valid argument that a system is adequately SAFE for a given application in a given environment”, whereby an argument is defined as “a connected series of claims intended to establish an overall claim.” In attempting to persuade others of the truth of an overall claim, we make supporting claims. These claims may themselves need further support. This gives rise to a hierarchy of claims (representing a logical chain of reasoning) by which an argument is established.

Given the multitude of generated safety-critical artifacts, let alone non safety-critical ones, it is clear that establishing this link through tracing alone is not possible, especially since simple traceability does not provide the expressiveness required for adequately describing the relations of various artifacts within a safety case context. This gives rise to the need for a dedicated safety case specific view on the artifacts.

6.1.1 Background and Motivation

“A safety case should communicate a clear, comprehensive and defensible argument that a system is acceptably SAFE to operate in a particular context.” [13]

The concept of the ‘safety case’ has already been adopted across many industries (including defense, aerospace, nuclear and railways), see [1], [34], [35], [36], [37] and has been emphasized in numerous research works such as [33], [38], [39], [40]. Studying the safety standards and guidance relating to these sectors, it is possible to identify a number of definitions of the safety case – some clearer than others. The definition given above attempts to cleanly define the core concept that is in agreement with the majority of the definitions we have discovered.

According to [13], a commonly observed failing of safety assessments is that the role of the safety argument is often neglected. In such safety cases, many pages of supporting evidence are often presented (e.g. hundreds of pages of fault trees or Failure Modes and Effects Analysis tables), but little is done to explain how this evidence relates to the safety objectives. The reader is often left to guess at an unwritten and implicit argument.

Both argument and evidence are crucial safety case elements that must go hand-in-hand. Argument without supporting evidence is unfounded, and therefore unconvincing. Evidence without argument is unexplained; it can be unclear how (or even if) safety objectives have been satisfied.

Safety cases thus have to be supported by the SAFE Meta-model, with the following rationale:

- The structured information management can be used as part of a safety argument in a safety case, and gives support to systematic safety/reliability analysis.
- The ability to support a safety case at a software architecture description level is important since it addresses an expanding area of functionality where the complexity is high.
- Traceability between the safety case and the design information is made possible, facilitating the work of the safety engineer, i.e. identifying the right information.
- Facilitate the system development of safety critical systems, by providing a link to where in a safety argument a certain “Entity” or “Item” under change is used. (Impact analysis of safety related systems)

6.2 Safety Case Description Elements and Relations

Within this section the description of the safety case description elements we use from GSN and their relations is given.

In order that safety cases can be developed, discussed, challenged, presented and reviewed amongst stakeholders, and maintained throughout the product lifecycle, it is necessary for them to be documented clearly. The documented argument of the safety case should be structured to be comprehensible to all its stakeholders. It should also be clear how the evidence is being asserted to support this argument.

GSN has been standardized into a first version since November 2011 and the specification document is readily available for free download. It is the primary source of the following two subsections.

The following subsection 6.2.1 gives a description of the standard elements used to describe artifacts while the subsequent section 6.2.2 details the types of relationships and the possible combinations used. Figure (4) depicts an example of the use of GSN [12].

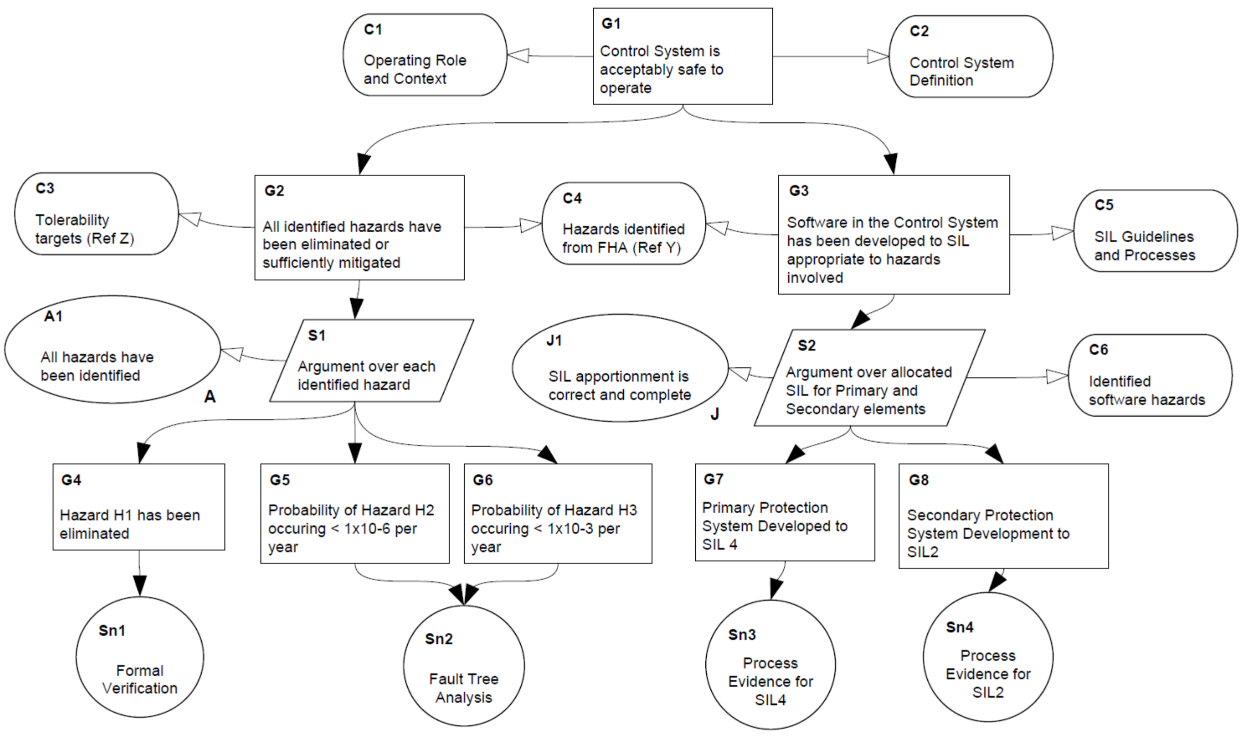
C1
Operating Role and Context
G1
Control System is acceptably safe to operate
C2
Control System Definition
C3
Tolerability targets (Ref Z)
G2
All identified hazards have been eliminated or sufficiently mitigated
C4
Hazards identified from FHA (Ref Y)
G3
Software in the Control System has been developed to SIL appropriate to hazards involved
C5
SIL Guidelines and Processes
A1
All hazards have been identified
A
S1
Argument over each identified hazard
J1
SIL apportionment is correct and complete
J
S2
Argument over allocated SIL for Primary and Secondary elements
C6
Identified software hazards
G4
Hazard H1 has been eliminated
G5
Probability of Hazard H2 occuring < 1x10-6 per year
G6
Probability of Hazard H3 occuring < 1x10-3 per year
G7
Primary Protection System Developed to SIL 4
G8
Secondary Protection System Development to SIL2
Sn1
Formal Verification
Sn2
Fault Tree Analysis
Sn3
Process Evidence for SIL4
Sn4
Process Evidence for SIL2

Figure 4: Example Goal Structure

6.2.1 Specification Elements

GSN [12] uses the following basic elements to describe the role of artifacts within a safety case:

Goal

- rendered as a rectangle,
- presents a claim forming part of the argument.

- One or more sub-goals may be declared for a given goal. This structure then asserts that if the claims presented in the sub-goals are true, this is sufficient to establish that the claim in the main goal is true.

Strategies

- rendered as a parallelogram,
- describe the nature of the inference that exists between a goal and its supporting goal(s)
- and are used to describe the nature of the inference which is asserted as existing between sub-goals and the parent goal.

Solutions

- A ***solution***, rendered as a circle, presents a reference to an evidence item or items.
- Multiple solutions may satisfy a goal.
- Multiple goals may be satisfied by one solution.

Contexts

- Claims can only be asserted to be true in a specified context. Context elements are used to make this relationship clear.
- A ***context***, rendered as an oblong rectangle with rounded out sides, presents a contextual artifact. This can be a reference to contextual information, or a statement.
- Where used, contexts define or constrain the scope over which the claim is made.

Assumptions

- An assumption is an intentionally unsubstantiated statement. The scope of an assumption is the entire argument. Having connected an assumption to a goal, the assumption is taken to be connected to the entirety of the argument supporting this goal.
- Therefore, it is not necessary to restate the assumption in the supporting argument.
- rendered as an oval with the letter ‘A’ at the bottom-right.

Justifications

- rendered as an oval with the letter ‘J’ at the bottom-right,
- presents a statement of rationale,
- and does not alter the meaning of the claim made in the goal, but provides rationale for its inclusion or its phrasing.
- Should an equivalent justification be required elsewhere in the argument, it will need to be re-stated or re-linked.

Furthermore it is possible to represent ***undeveloped entities***, rendered as a hollow diamond applied to the center of an element, indicating that a line of argument has not been developed. It can apply to goals (as below) and strategies.

A specific instance of ***undeveloped entities*** is an ***undeveloped goal***, rendered as a rectangle with the hollow-diamond ‘undeveloped entity’ symbol at the center-bottom, presenting a claim which is intentionally left undeveloped in the argument.

6.2.2 Relations and possible expressions

The core elements described in the previous section are linked using the following types of relationships [12]:

SupportedBy

- rendered as a line with a solid arrowhead, allows inferential or evidential relationships to be documented.
- Inferential relationships declare that there is an inference between goals in the argument. Evidential relationships declare the link between a goal and the evidence used to substantiate it.
- Permitted connections are:
  - goal-to-goal,
  - goal-to-strategy,
  - goal-to-solution,
  - strategy to goal.

InContextOf

- rendered as a line with a hollow arrowhead, declares a contextual relationship.
- Permitted connections are:
  - goal-to-context,
  - goal-to-assumption,
  - goal-to-justification,
  - strategy-to-context,
  - strategy-to-assumption
  - and strategy-to-justification.

6.2.3 Steps for building a safety case goal structure

Works of Kelly and McDermid as well as the GSN standard base on well-revised argumentation works and suggest a multi-step approach to formulating goal structures and hence building safety cases, as seen in the following subsections.

Top Down: When classically working from Safety Goals

For a classic staged approach to safety arguments, Kelly [5] defines six steps in the top-down development of a goal structure:

1. Identify the goals to be supported;
2. Define the basis on which the goals are stated;
3. Identify the strategy used to support the goals;
4. Define the basis on which the strategy is stated;
5. Elaborate the strategy (and proceed to identify new goals – back to step 1), or step 6;

6. Identify the basic solution.

Bottom-Up: When working from existing evidence

When analyses evidence already exists or when an existing safety case needs updating, it is possible to adapt Kelly's six steps (see Section 2.3) for top-down GSN development, into a process which can be used to develop a goal structure from the bottom up [12]:

1. Identify evidence to present as GSN solutions;

2. Infer 'evidence assertion' claims to be directly supported by these solutions, and present these as GSN goals;

3. Derive higher-level sub-goals that are supported by the evidence assertions;

4. Describe how each layer of sub-goals satisfies the parent goal (i.e. strategy);

5. Check that any necessary contextual information is included;

6. Check back down the structure for completeness;

7. Join the resulting goal structure to a known top goal or a set of sub-goals.

Using these methods in conjunction with the steps explained in section [8.3], it is possible to generate and document safety cases.

6.3 Safety Case Patterns

The idea of design patterns was original proposed by an architect [15] who wrote several books on the field of urban planning and building construction. The concept of design patterns is a universal approach to describe common solutions to widely recurring design problems. A pattern is an abstract representation for how to solve a general problem which occurs over and over in many applications. Describing proven solutions as patterns provides a good documentation for these solutions and makes them more accessible for future use in new systems. Ever since, this concept has been applied to many different domains including hardware and software design.

Patterns support and help designers and system architects choose suitable solutions for design problems, and thus also found resonance in the development of safety-critical embedded systems, where a high level of confidence in the implemented solutions (in this case through repeated testing and use in field) is highly desirable. Reusing proven solutions also makes it possible to reuse the arguments originally used to prove the safety of these solutions and thus gives rise to the concept of safety case patterns, as discussed in [14] and [10]. An approach to combine arguments by the compositional use of patterns has been suggested in [16].

6.3.1 Introduction

It is easy to imagine a situation where a developer has already implemented a function many times and reused many if not all of its elements wanting to also reuse the safety case used to argue the development's safety. In fact, informal reuse of safety case material is already commonplace, especially within stable and well understood domains. However, this type of uncontrolled and often ad hoc reuse can fail to fully exploit opportunities for reuse, and can in some cases be potentially dangerous [14]. This is mainly due to:

- Artifacts being reused inappropriately
- Reuse occurring in an ad-hoc fashion
- Loss of knowledge

- Lack of Consistency / Process Maturity
- Lack of traceability

These problems stem from the general issue of documentation, especially when it is informal and textual, as is most documentation. Using some kind of structured (if not formalized) description, it is possible to reuse the safety cases fully or partially using the concept of patterns.

A deeper view into pattern use in software development in general and in safety-critical systems specifically is provided in the following subsections.

As with all patterns, a catalogue of mandatory and optional features will have to be established, examples for which can be found in [14]. A suggestion for a pattern template for safety-critical application was also provided in [32].

A key element of the pattern format when applied to safety cases is the notion of pattern applicability. Applicability defines under what circumstances the pattern can be legitimately applied. For example, descriptions of applicability could indicate which standards the pattern adheres to, the level of design detail required or the assumption of system behavior. Applicability of a safety case pattern is perhaps more closely tied to the structural description of the pattern than with design patterns. The goal structure representation of the pattern may specifically require certain elements to be present in the goal structure into which the pattern is placed (e.g. a context entity, model, assumption or constraint).

A difference in emphasis between the design patterns of Gamma (widely used in architecture design) and safety case patterns is that, in addition to the pattern rationale being documented by intent and motivation, the rationale behind a safety argument or safety process should also be embedded in the elements of the structural description using the goal structuring notation (this is in our case achieved through the use of the GSN elements - strategies, assumptions and justifications).

Safety case patterns are intended to describe partial solutions and will not typically describe the complete structure of a system safety argument. It is expected that a collection of patterns will therefore emerge over time forming a ‘recipe book’ of safety arguments and processes, a number of which would be used together to aid in the construction of the safety case, as discussed in [14], [10], and [16].

6.3.2 Background and related work

While the Gang of Four book [17] has been the most popular work on design patterns over the last two decades, there have been several attempts in the literature to adapt this concept in many fields of system design. In 1987 Cunningham and Beck presented five patterns for designing windows-based user interfaces with Smalltalk [18]. Around the same time, Jim Coplien began to document C++ Idioms that represent specific constructs like patterns for C++. He published these idioms as a book in 1991 [19], and later he published these idioms and patterns as paper in 1997 [20].

From 1990 to 1993, several papers addressing the use of design patterns in object oriented programming were published at the OOPSLA (Object-Oriented Programming Systems, Languages, and Applications) conference. In 1994, the Hillside group (see http://hillside.net) organized the first PLoP (Pattern Languages of Programs) annual conference. The revised papers from PLoP are normally published in the book series “Pattern Languages of Program Design” (see e.g. [21]). Meanwhile, Buschmann et al. published the book “Pattern-oriented software architecture: a system of patterns” [22] which includes a collection of relatively independent solutions to common design problems represented as a catalog of design patterns.

Another well-known work is the catalog presented by Bruce Douglass in [23]. This catalog includes a set of patterns for real-time embedded systems. The presented patterns deal with real-time design issues like concurrency, resource sharing, distribution, and SAFE and reliable architecture.

The concept of design pattern has become an important area of research in many fields like: fault tolerance [24], telecommunications [25], embedded systems [26], [27], security [28], [29], and many other fields. Each of these fields has its own patterns and sometime its own representation, but all follow the basic principle of design patterns. Works seeking to structure and facilitate the selection of architectural patterns and measures adequate for safety-critical requirements have already been presented [32], but the selection process is still highly manual and case dependent, and focuses on individual patterns as a guideline with no link to a holistic view.

On the other hand, there is extensive work on formalized reasoning about safety cases and safety concept argumentation [38], [42], [13], as well as the use of patterns for safety cases [14], which culminated in a standardization of the structured notation known as GSN [12]. An analysis of industrial safety cases yielded a finite set of patterns [41] from which it is possible to construct a pattern library of problem types and their respective solutions and use it to automate the generation of and argumentation about safety concepts as well as safety cases, for which it is planned to implement a proof-of-concept in the research CASE tool AutoFOCUS3 [30] in the next project phase. Safety case pattern catalogues and the capability for compositional argumentation is suggested in several research works such as [6], [7], [8], [9], [10], [11] and [16].

6.3.3 Safety case pattern templates

Safety case templates can be generated, stored and reused for many categories of patterns, such as:

- Strategy (Problem solving) patterns
- Design patterns
- Constraint patterns and Safety Mechanisms
- COTS – Commercial Off The Shelf Components

Problem solving strategy patterns

As discussed in [41] the range of problems developers in a domain face and subsequently the algorithms they favor in solving them are not unlimited. By identifying recurring paradigms and analyzing them it is possible to generate templates of their safety cases. The strategy node is the core element of the actual argumentation in a problem pattern safety case template, supporting the justification of goals being fulfilled by sub-goals. By applying strategy patterns, it is possible to build argumentations using only accepted justifications. This way confidence in the correctness of the argument can be increased. Still the appropriateness and correct use of a pattern has to be evaluated before trusting a safety case, but the question of the fundamental validity of each single argumentation step itself need not be argued. On the long run, it is desirable to establish general and domain-specific patterns as a pattern library for a faster and easier creation of safety arguments [41].

<u>Example Pattern: Logical transformation</u>

The goal is a logical combination leading to desirable or undesirable situations. It is possible to generate a safety case skeleton that represents the pattern of avoiding a situation in which the constraint is violated, e.g. to keep acceleration larger or equal than a lower bound in an adaptive cruise control. This is simplified by transforming the goal into avoiding an acceleration which is smaller than the lower bound [41], as shown in figure (5).

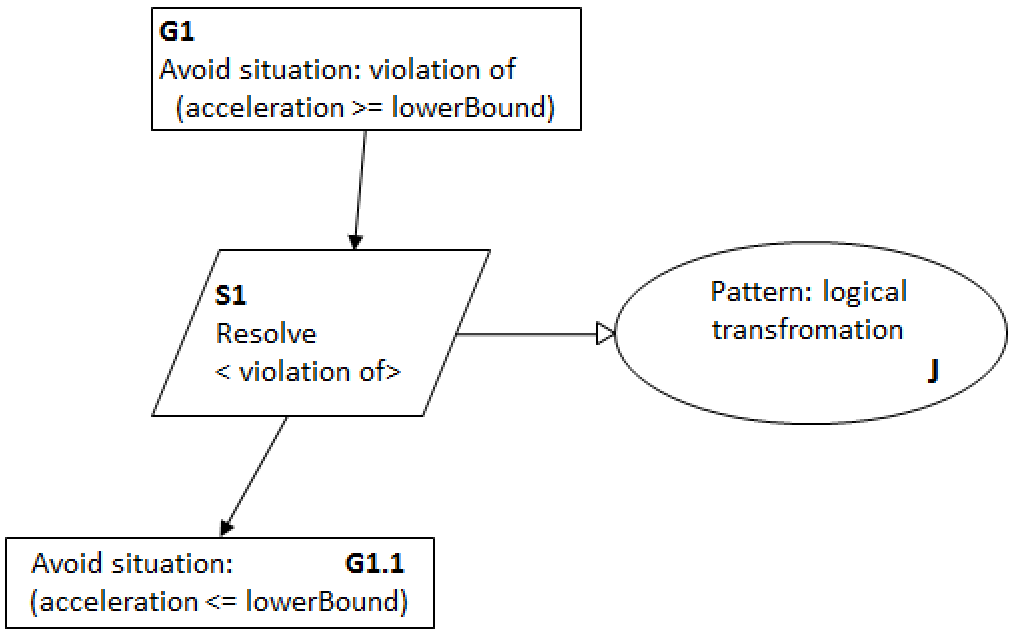
G1
Avoid situation: violation of (acceleration >= lowerBound)
S1
Resolve < violation of>
Pattern: logical transfromation
J
Avoid situation: (acceleration <= lowerBound)
G1.1

Figure 5: An example usage of the logical transformation pattern

Design Patterns

ISO 26262 [1] requires that component architectures, independently of their functionalities, display certain characteristics or adhere to constraints. Some of these such as modularity, simplicity, and an adequate level of granularity and encapsulation are simply good engineering practices aimed at avoiding failures arising from unnecessary complexity.

Other requirements, such as freedom from interference, which is the absence of dependent failures (cascading and common cause failures) in safety-critical components, are aimed at guaranteeing correct operation of safety-critical functions.

Furthermore, the ISO26262 [1] dictates that "if the embedded software has to implement software components of different ASILs, or safety-related and non-safety-related software components, then all of the embedded software shall be treated in accordance with the highest ASIL, unless the software components meet the criteria for coexistence in accordance with ISO 26262-9:2011, Clause 6." This latter clause mentions freedom from interference and component independence as being the central requirements for such co-existence.

According to ISO26262, there are several ways for components to have freedom from interference, such as:

- be functionally diverse (the use of totally different approaches to achieve the same results);
- be based on diverse technologies (the use of different type of equipment to perform the same result)
- not share common parts or services whose failure could result in a dangerous mode of failure of all systems
- be designed so that the predominant failure mode for common support systems (e.g. power supply) is in a SAFE direction (i.e. fail-safe)
- not share common operational or maintenance or test procedures

- be physically separated such that foreseeable failures do not affect redundant safety-related systems.

Example Pattern: Redundancy

The guidelines above have in turn matured into various design patterns, many of which revolve around redundancy and partitioning, which is the separation of functions or component elements to achieve a design, which can be used for fault containment to avoid cascading failures. The design patterns vary in their addressed context, structure, and presented solution and can be categorized in many ways. For example for redundancy there exist [32]:

- **Hardware Patterns**: Includes the patterns that contain explicit hardware redundancy. This group contains the following patterns:
  - Homogeneous Duplex Pattern
  - Heterogeneous Duplex Pattern.
  - Triple Modular Redundancy Pattern
  - M-Out-Of-N Pattern
  - Monitor-Actuator Pattern
  - Sanity Check Pattern
  - Watchdog Pattern
  - Safety Kernel Pattern

- **Software Patterns**: Includes the patterns that use software diversity (redundancy) to tolerate software faults. This group contains the following patterns:
  - N-Version Programming Pattern
  - Recovery Block Pattern.
  - Acceptance Voting Pattern
  - N-Self Checking Programming Pattern
  - Recovery Block with Backup Voting Pattern

- **Combination of Hardware and Software Patterns**: Include the following patterns that do not contain explicit hardware redundancy or software diversity:
  - Protected Single Channel Pattern
  - 3-Level Safety Monitoring Pattern

All the above patterns fall under one general paradigm and as such can be covered by a general safety case pattern template as shown in figure (6), namely that of solving the lack of confidence in the safety-critical channel using a redundancy strategy. This general template can in turn be instantiated and subsequently further specified and extended to suit the specific case and used solution, for example using context elements.

Decision trees to provide assistance in selecting the suitable pattern have been developed and discussed in [32].

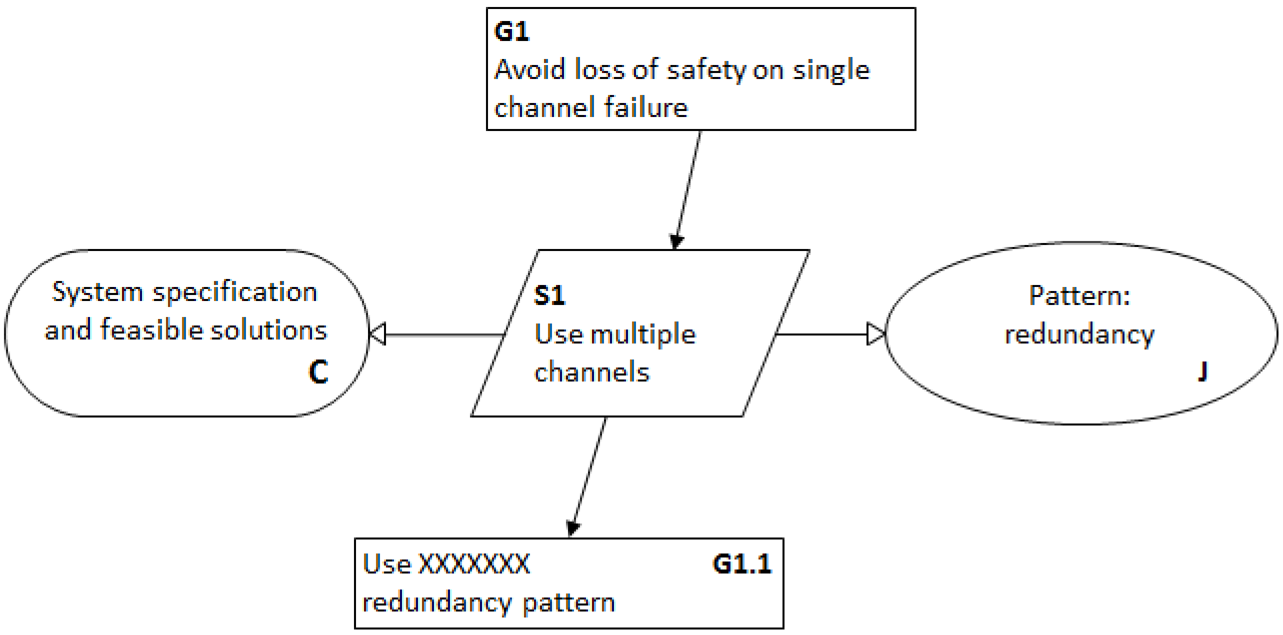
G1
Avoid loss of safety on single channel failure
System specification and feasible solutions
C
S1
Use multiple channels
Pattern: redundancy
J
Use XXXXXXX redundancy pattern
G1.1

Figure 6: Safety case template for channel redundancy patterns

Safety Mechanisms

Safety mechanisms are trusted solutions to repeated engineering problems facing designers and developers of safety-critical systems which have developed to patterns commonly used in the realization of technical safety concepts. A collection of safety mechanisms has been discussed in [43]. Safety mechanisms, their integration into the SAFE Meta-model and the generation of their code have been covered in WT3.6. Similarly to Strategy patterns and design patterns, safety mechanisms can also be accompanied by safety case snippets or templates that can be instantiated as needed and integrated into the safety case at the generation point.

COTS Components

These pattern category targets the re-use of previously developed hardware and software components, including COTS products. The safety cases for these may be built around proofs of validity or the proven in use argumentation in case of sufficient field data. Similarly to safety mechanisms, COTS components can also be accompanied by safety case snippets or templates that can be instantiated as needed and integrated into the safety case at the insertion point of the COTS product.

6.3.4 Using safety case patterns

Combining the parts explained in the previous subsections a 5-step approach could be suggested using which it would be possible to quickly construct safety cases from a pattern library.

Step 1: Definition of Safety Goal and identification of problem type

Step 2: Selection of suitable paradigm or design pattern / safety mechanism / COTS Component

Step 3: Creation of safety goal instances and generation of safety case skeleton

Step 4: Integration into existing safety cases (with compositional argumentation)

Step 5: Testing for safety case / concept completeness and consistency

6.4 Testing of safety cases

Safety cases can have more than one role, depending on when and how they are generated and used. Each of these roles defines what types of tests can be performed and to which means.

Descriptive Safety Cases

Use:

When goal structures are generated to argue about existing products or during products development a safety case evolves which describes the safety of the system under development and the fulfillment of its safety goals. This safety case describes the status quo and aims at providing a clear and justifiable argument that the safety goals are met.

Testing:

Using the set of relation rules identified in Section 6.2.1 and 6.2.2, e.g., each **Goal** must resolve to a **Solution**, in a model-based development environment it is possible to perform completeness and consistency checks on this type of safety case.

Prescriptive Safety Cases

Use:

In conjunction with the use of patterns, once a decision is made to solve a known problem or use a trusted solution pattern or purchase a COTS component, the developer can directly create an instance of the (partial) safety case template or skeleton for the known pattern/component. It thus becomes possible to know what the safety case *should* look like and to use this as development guideline.

Testing:

Using the generated safety case skeleton as a reference, the developer can test existing solutions against it to see whether their arguments hold.

The extent of the testing depth and automation rely on how high the integration of the safety case elements into the development artifact landscape is.

7 Safety Cases with EAST-ADL

Within this section the current status of the architecture description language EAST-ADL V2.1 with regard to safety cases is described. Furthermore, proposals for an extension of the EAST-ADL concept are described which could lead to an enhancement of the possibility to model and document safety cases according to ISO 26262.

EAST-ADL introduces different levels of abstraction within a hierarchical modeling concept which facilitates controlling the complexity of systems. These levels are:

- Vehicle level,
- Analysis level,
- Design level,
- Implementation level, and
- Operational level.

7.1 Current status of EAST-ADL and other suggested extensions

Besides the different abstraction levels EAST-ADL includes several packages like, for instance, the variability package, the timing package, and the dependability package which is of special interest for this work task. An overview on the dependability package [44] is given in figure (6).

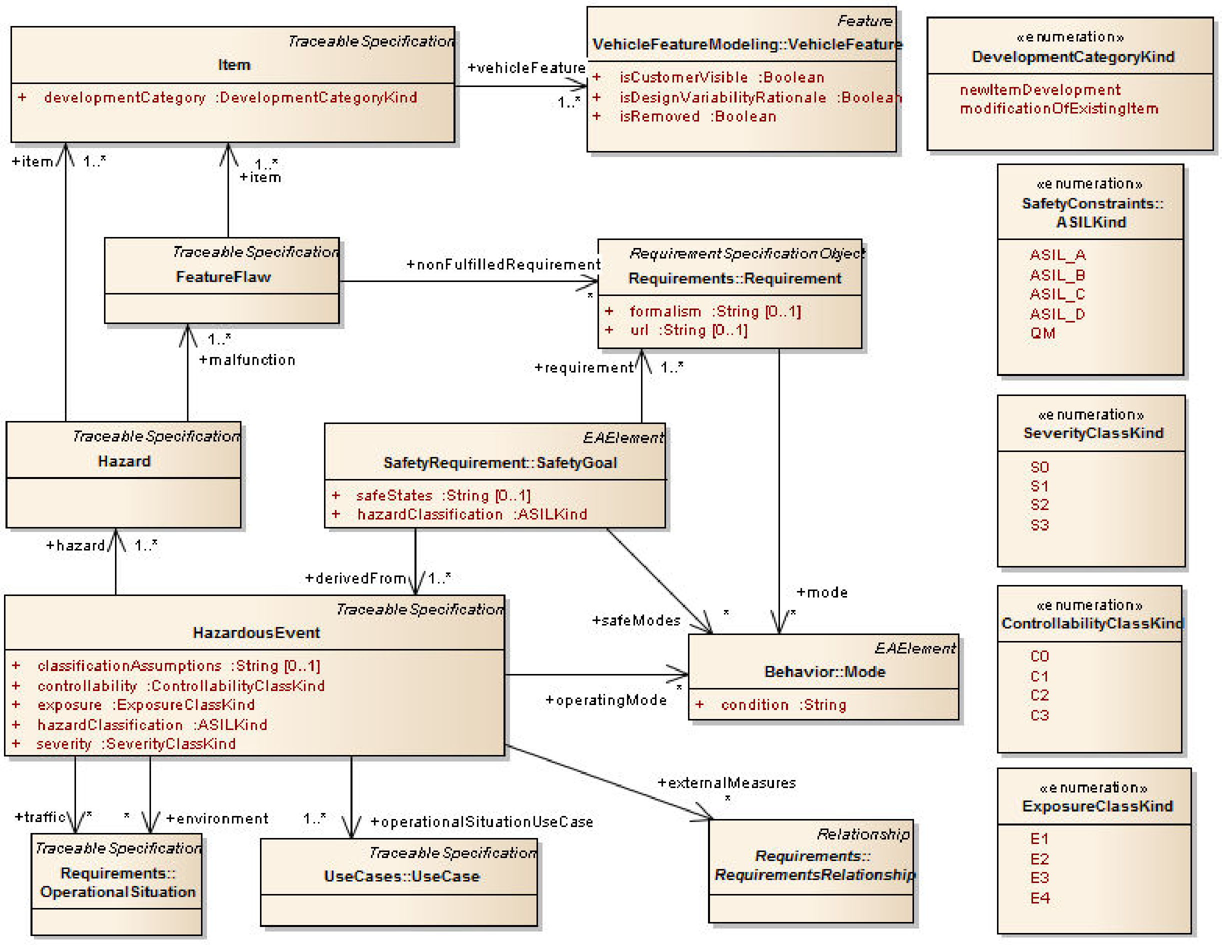

Figure 7: EAST-ADL Dependability Package

As it can be seen in figure (6), the basic artifacts needed for expressing safety activities, like for instance hazards, hazardous events and safety goals, are already included. For WT 3.1.3 it should be the objective to reuse as much as possible the already existing content provided in EAST-ADL.

Furthermore, EAST-ADL (version 2.1.9.1) includes Safety Case description capability in the SafetyCase Sub-Package [44], shown in figure (8).

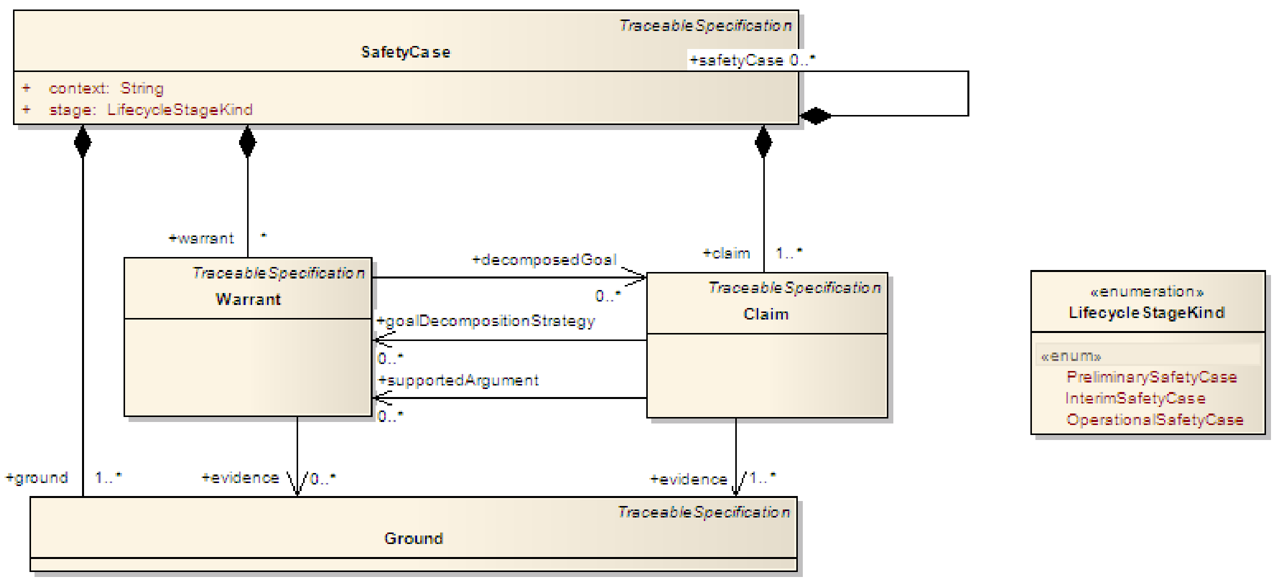

Figure 8: EAST-ADL Safety Case

These various elements are described in the following subsection [EAST-ADL Domain Model Specification 2.1.9.1)] and their relations shown in detail in figure (8):

Claim:

A Claim represents a statement, the truth of which needs to be confirmed and which has associations to the strategy for goal decomposition and to supported arguments. It also holds associations to the evidences for the SafetyCase.

Ground:

Claim is based on Grounds (evidences) - specific facts about a precise situation that clarify and make good the Claim.

Ground represents statements that explain how the SafetyCase Ground clarifies and make good the Claim.

Ground has associations to the entities that are the evidences in the SafetyCase.

LifecycleStageKind:

The SafetyCase should be initiated at the earliest possible stage in the safety program so that hazards are identified and dealt with while the opportunities for their exclusion exist.

The LifecycleStageKind is an enumeration meta-class with enumeration literals indicating safety case life cycle stage.

SafetyCase:

SafetyCase represents a safety case that communicates a clear, comprehensive and defensible argument that a system is acceptably SAFE to operate in a given context.

Safety Cases are used in safety related systems, where failures can lead to catastrophic or at least dangerous consequences.

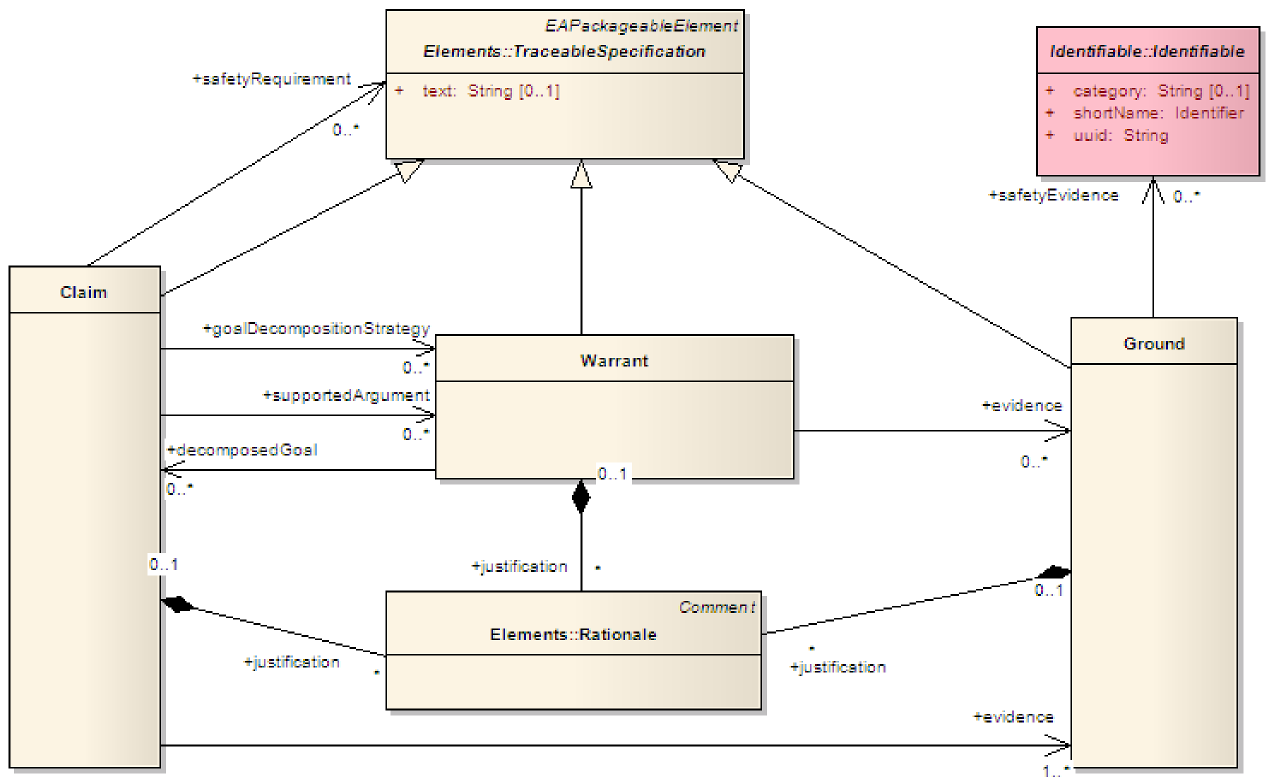

Figure 9: EAST-ADL Ground, Warrant and Claim Diagram

The structures shown in figure (7) is in turn based on the Toulmin Model of Argumentation defined in Stephen Toulmin's 1958 work "Uses of Argumentation" [2], the main concept of which is shown graphically in figure (8).

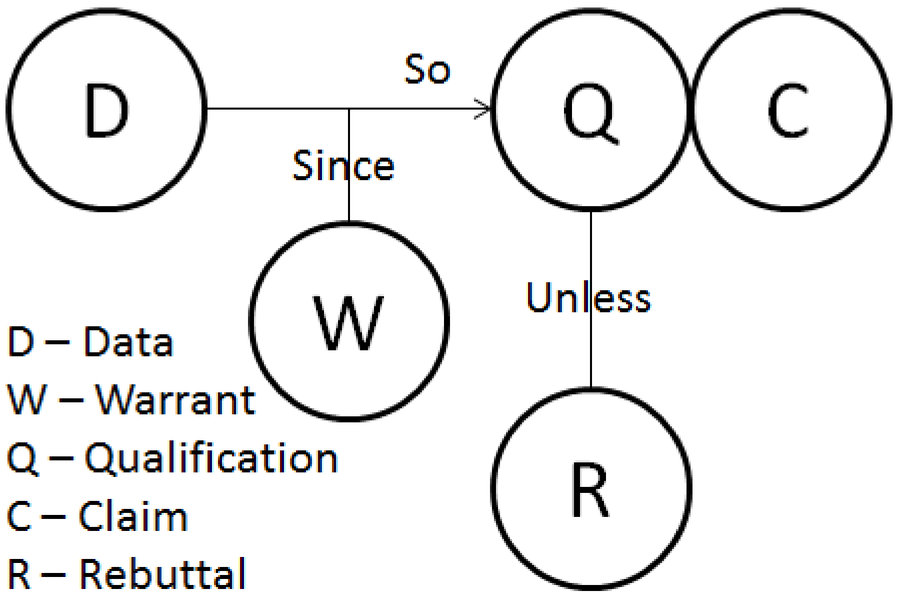

Figure 10: Toulmin Argumentation Concept Elements

We have chosen to expand this capability using GSN, to specifically offer a few extra elements such as context, assumption and justification elements, as will be seen in the following section.

7.2 Proposed extensions to EAST-ADL

Investigating the current capability to express safety cases in EAST-ADL, as shown in the previous section 7.1, showed that there is a potential need for extensions, according to the description given in Section 6. Our research into safety case modeling has favored the Goal Structuring Notation (GSN), which was introduced in Section 6, and has already been successfully used in the nuclear, aerospace and railway domains. Extension of EAST-ADL to include Elements of GSN has already been suggested in several ongoing research projects, such as the proposed safety case extensions for EAST-ADL2 in the ATTEST project. The approach followed in this project is most similar to our exploration and findings and will be used here where suitable to avoid repetition. The potential extensions together with their rationale are described in the following.

Introduction of Safety Case Class with GSN Notations

Instead of using the current SafetyCase we support the extension of EAST-ADL with a **Safety Case** class based on GSN as suggested in [3] and shown in figure (9).

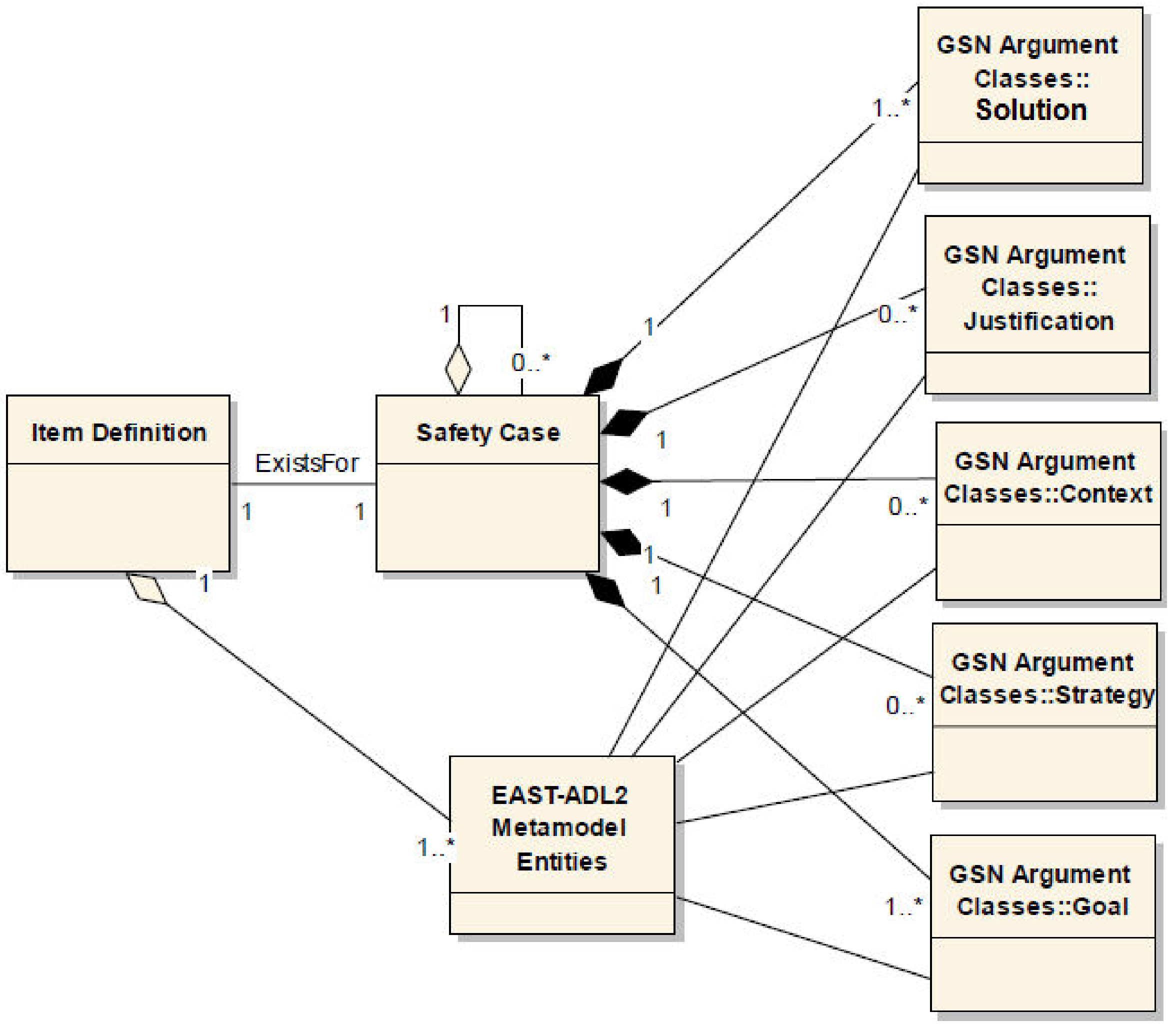
GSN Argument Classes:: Solution
GSN Argument Classes:: Justification
GSN Argument Classes::Context
GSN Argument Classes::Strategy
GSN Argument Classes::Goal
Item Definition
ExistsFor
Safety Case
EAST-ADL2 Metamodel Entities
1
0..*
1..*

Figure 11: Proposed EAST-ADL2 extension for modeling safety cases in ATESST Project

The Safety Case class is centered on the Safety Goal, which can be decomposed directly into 2 or more goals or indrectly via the use of a strategy. Each goal shall resolve to at least one solution. This rule can be used later in section 6.4 to test safety cases for completeness and consitency. Contexts and justification are presented and assumptions can be included as well, as seen in the internal class diagram, shown in figure (10) with the safety goal forming the center of the safety case structure. The elements and relations shown in figure (10) were previously explained in section 6.2.

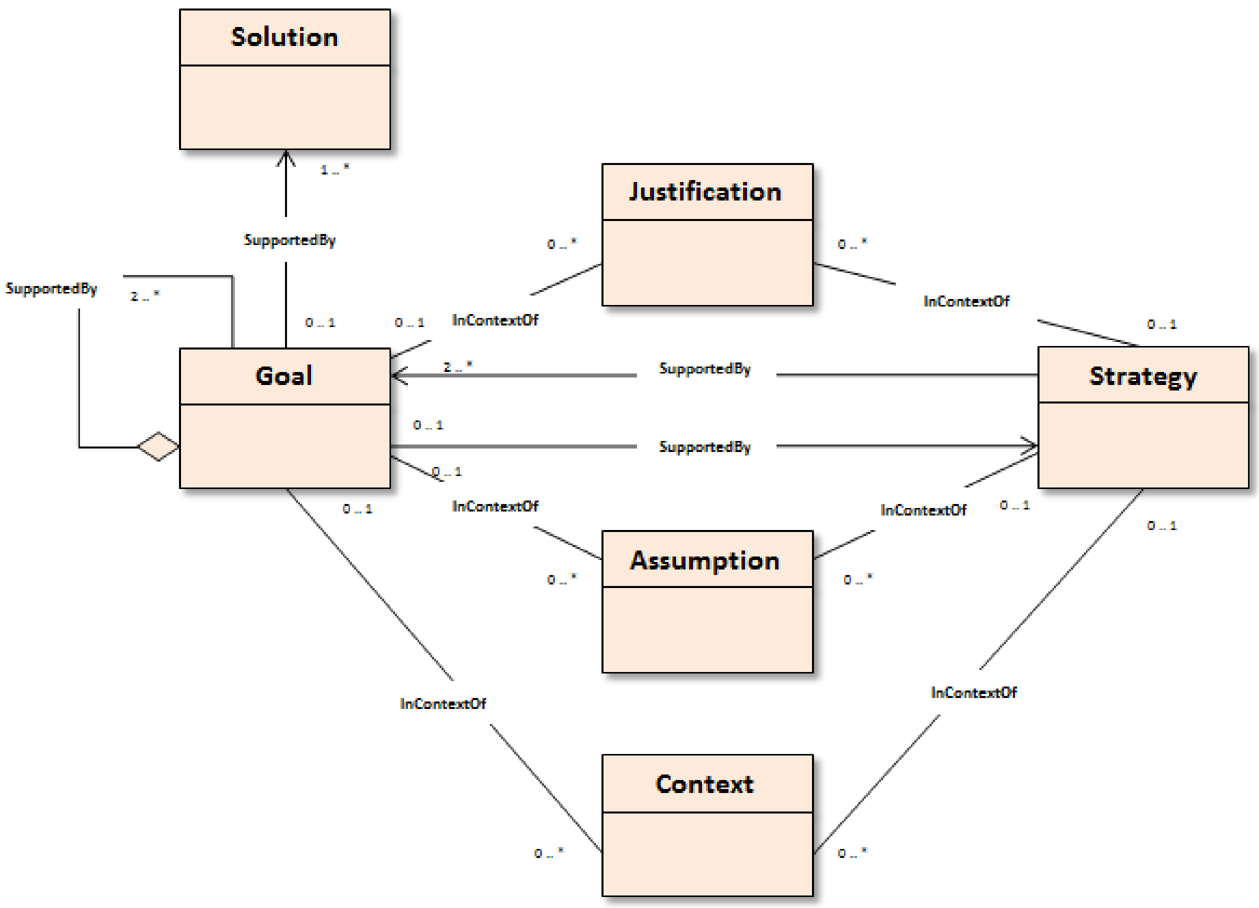

Figure 12: Internal Structure of proposed Class Safety Case

Linking Safety Case Class to EAST-ADL Models

An especially important element of the class Safety Case is the **Solution** entitity, which represents any information that supports or, in its ultimate form, proves that the Goal it is connected to is achieved. As such, the information can be of many types. In figure (11) a class diagram of **Solution** is presented which shows how the **Solution** entity can be specialized to hold the wide array of information that can support a claim. The second level in the hierarchy can consist of general EAST-ADL classes that could supply this information, as shown.

The safety case argument can be seen as consisting of two general branches; the product safety argument and the SAFE process argument. The later part of a safety case argument is considered to be out of scope of the EAST-ADL metamodel since it is supposed to be independent of methodology. However, such process parts as are covered in activities of WP6 can be used in this argumentation as well, i.e. support of assessment activities and application rules etc.

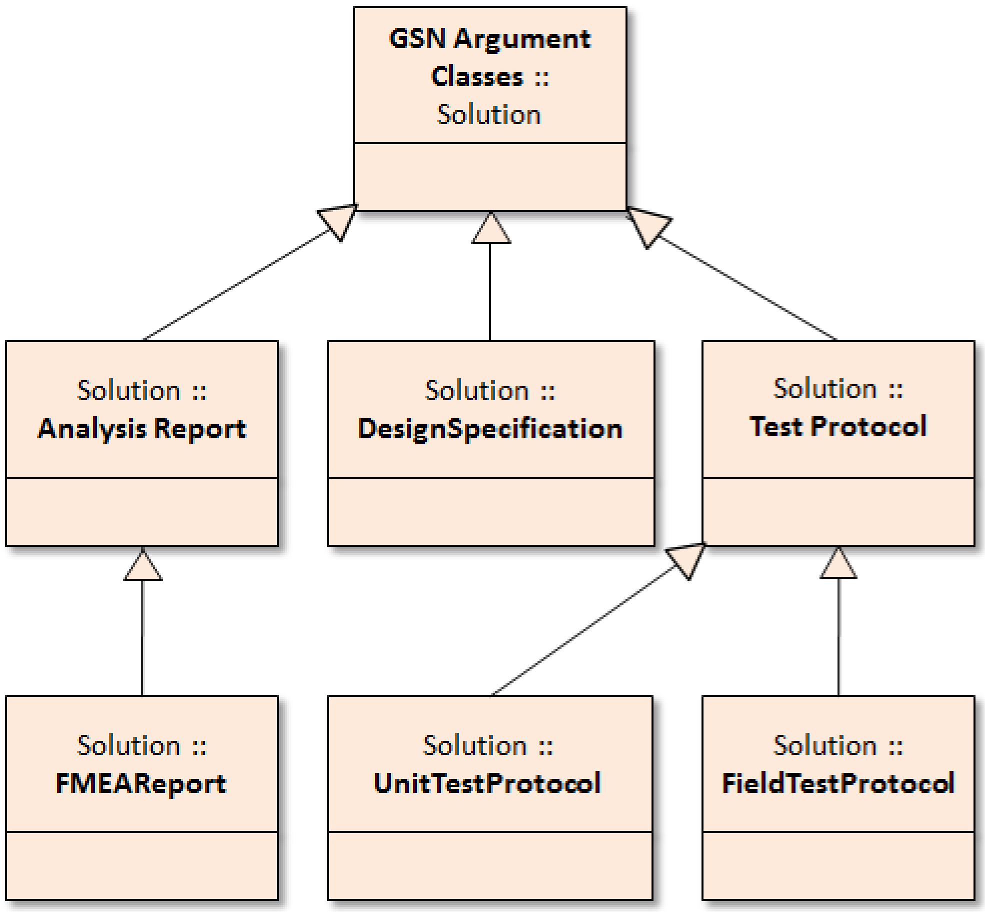
GSN Argument Classes :: Solution
Solution :: Analysis Report
Solution :: DesignSpecification
Solution :: Test Protocol
Solution :: FMEAReport
Solution :: UnitTestProtocol
Solution :: FieldTestProtocol

Figure 13: Possible links from EAST-ADL to Solution Element

There are several alternatives to integrate the safety case package to the EAST-ADL Meta-model:

1- A collection of EAST-ADL entities are associated with the safety case entity.

2- EAST-ADL2 entities are directly associated with the safety case entity.

3- GSN entities are directly associated with EAST-ADL entities.

While the first two alternatives seem easier the traceability they allow is also much limited. Traceability is a central as well as mandatory aspect of safety-critical development and as such we suggest following alternative 3 were possible.

A potential first instance of this alternative would be link many of the output results of the EAST-ADL V&V (verification and validation) package to the Safety Case Solution element, as shown in figures (9) and (11).

8 WT 3.1.3 Contribution to SAFE Meta-model

Within this section the contribution of WT 3.1.3 to the SAFE Meta-model is described. At the beginning an overview about the model is given which is followed by the detailed description of the classes and interconnections for the simple class option.

8.1 Overview

The contribution of WT 3.1.3 is mainly captured in two class diagrams of the SAFE Meta-model created in Enterprise Architect. In the first diagram, which is shown in figure (14), the artifacts needed for the hazard analysis and risk assessment and their interconnections are modeled [46]. The attributes shown in this diagram are only those that are not included in the referenced classes of the current version of EAST-ADL.

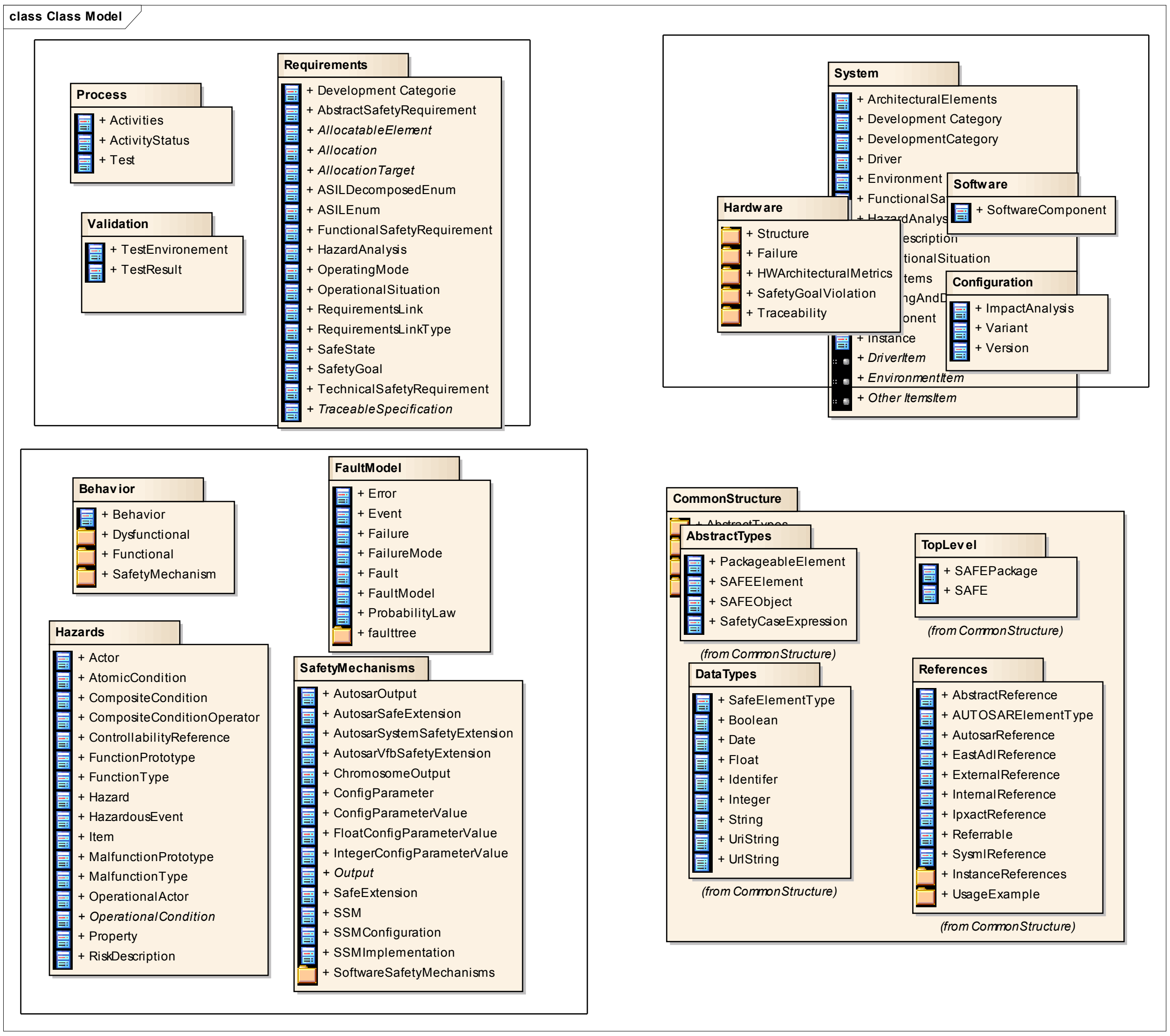

Figure 14: SAFE Meta-model Overview with SafetyCaseExpressions highlighted

The relation of SafetyCaseExpressions [46] to the SAFE Meta-model Elements is shown in the following figure.

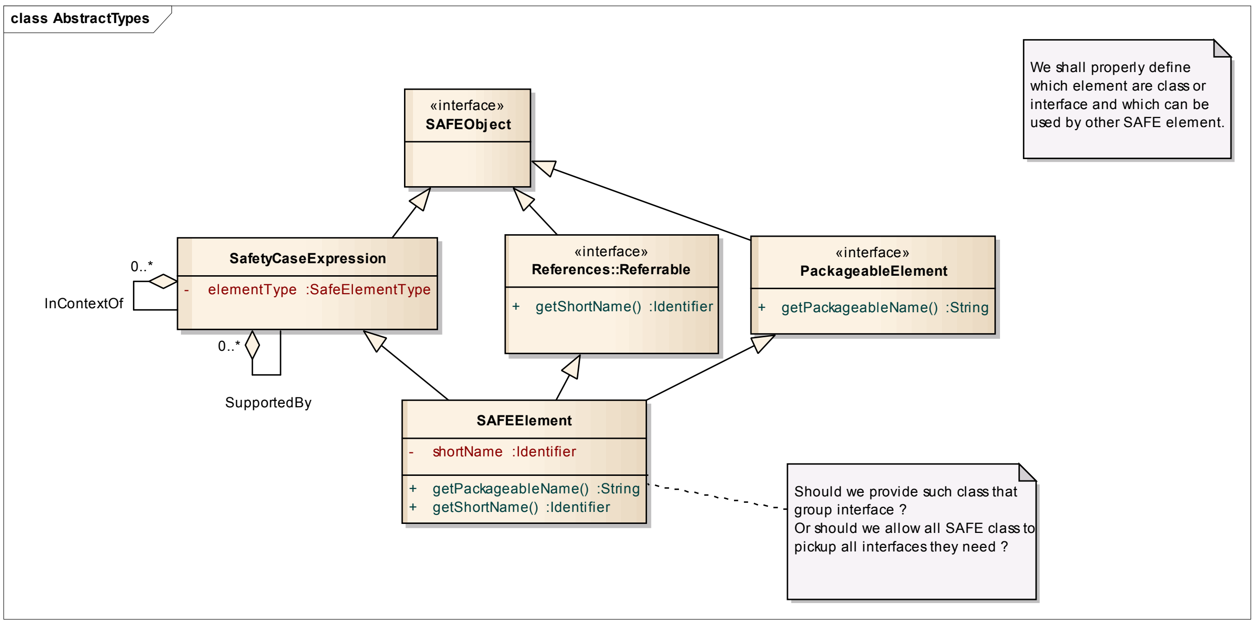
class AbstractTypes
«interface»
SAFEObject
We shall properly define which element are class or interface and which can be used by other SAFE element.
SafetyCaseExpression
- elementType :SafeElementType
0..*
InContextOf
0..*
SupportedBy
«interface»
References::Referrable
+ getShortName() :Identifier
«interface»
PackageableElement
+ getPackageableName() :String
SAFEElement
- shortName :Identifier
+ getPackageableName() :String
+ getShortName() :Identifier
Should we provide such class that group interface ?
Or should we allow all SAFE class to pickup all interfaces they need ?

Figure 15: Overview of SafetyCaseExpression relations to SAFE Object in SAFE Meta-model

As it can be seen there are various elements originating in the current EAST-ADL version that can be reused for the SAFE Meta-model. In case of referencing an element it is assumed that the attributes defined for the class in EAST-ADL are inherited. A decision of whether to extend the SafetyCaseExpressions Class (and hence the SAFE Meta-model) with the subclasses shown in Section 7.2 will be made at the next phase in the project, with the stabilization of the SAFE Meta-model.

8.2 Class Description

SAFEElement

Database: Java, *Stereotype: , Package: AbstractTypes*

Notes: base class for all SAFE class that represent something (i.e. not technical class).

SafetyCaseExpression

Database: Java, *Stereotype: , Package: AbstractTypes*

Notes: *provide information like justification or explanation on a specific element in safety case .*

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	elementType	SafeEl-ementType			0	0	0		

Elementtype can be one of six GSN types.

1- Goals

2- Strategies

3- Solutions

4- Contexts

5- Assumptions

6- Justifications.

These can linked to each other by 2 types of connections:

1- SupportedBy

2- InContextOf.

Relationships

Columns	**Association**	**Notes**
	SafetyCaseExpression. SAFEObject.	
	SAFEElement. SafetyCaseExpression.	
InContextOf	**0..*** SafetyCaseExpression. SafetyCaseExpression.	part of GSN interface
SupportedBy	**0..*** SafetyCaseExpression. SafetyCaseExpression.	part of GSN interface

8.3 Proposed Interaction with SAFE Meta-model Elements for Documentation

Aside from being useful for developing SAFE systems, a safety case's original and ongoing purpose is to document the correct development of the safety-critical product and the fulfillment of all safety goals.

By assigning the correct safety case element to each used artifact and joining the artifacts in the safety case context through the appropriate relations it is possible to generate safety case reports encapsulating the information required for proving fulfillment of the safety goals in a comprehensible and defensible manner. The depth of the reports and the degree of automation depends on the level of integration of the safety case elements into the generated Meta-model artifacts as shown in section 7.2. In the following it is explicitly not recommended using GSN on an "atomic level" on individual SAFE Meta-model artifacts, but rather to use GSN to explain the structure of a safety case report, based on the concepts of the SAFE Meta-model.

In [5] the following areas of a safety case report are proposed:

- Scope
- System Description
- System Hazards
- Safety Requirements

- Risk Assessment
- Hazard Control / Risk Reduction Measures
- Safety Analysis / Test
- Safety Management System
- Development Process Justification

The last two areas are out of the scope of this concept because the safety management system and the development process specification are not in the scope of the SAFE project.

Figure (16) illustrates the strategies of a safety case report, which follows the following structure and exploits concepts defined in the SAFE Meta-model.

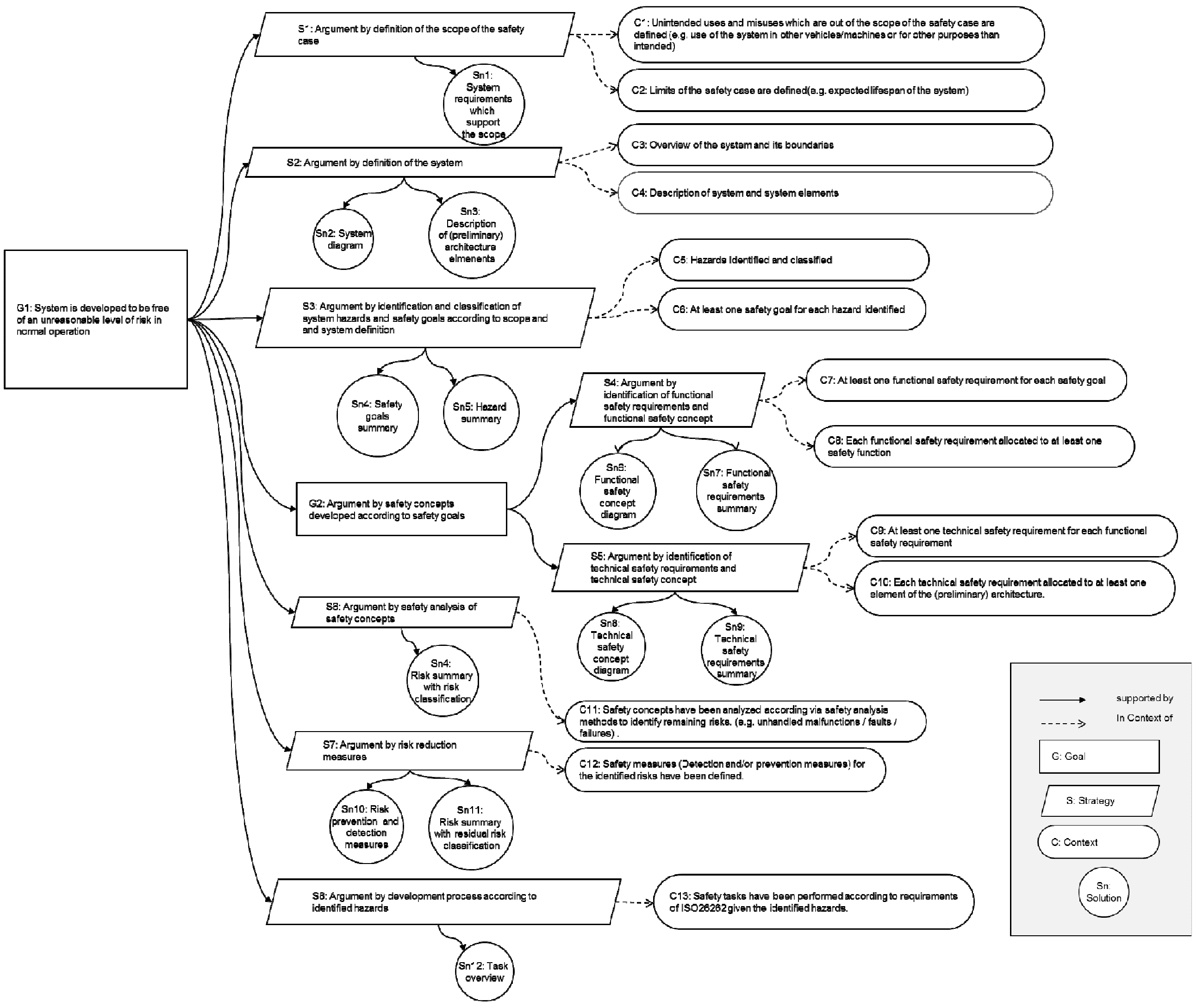
G1: System is developed to be free of an unreasonable level of risk in normal operation
S´: Argument by definition of the scope of the safety case
Sn1: System requirements which support the scope
C´: Unintended uses and misuses which are out of the scope of the safety case are defined (e.g. use of the system in other vehicles/machines or for other purposes than intended)
C2: Limits of the safety case are defined(e.g. expected lifespan of the system)
S2: Argument by definition of the system
C3: Overview of the system and its boundaries
C4: Description of system and system elements
Sn2: System diagram
Sn3: Description of (preliminary) architecture elmenents
S3: Argument by identification and classification of system hazards and safety goals according to scope and and system definition
C5: Hazards identified and classified
C6: At least one safety goal for each hazard identified
Sn4: Safety goals summary
Sn5: Hazard summary
G2: Argument by safety concepts developed according to safety goals
S4: Argument by identification of functional safety requirements and functional safety concept
C7: At least one functional safety requirement for each safety goal
C8: Each functional safety requirement allocated to at least one safety function
Sn6: Functional safety concept diagram
Sn7: Functional safety requirements summary
S5: Argument by identification of technical safety requirements and technical safety concept
C9: At least one technical safety requirement for each functional safety requirement
C10: Each technical safety requirement allocated to at least one element of the (preliminary) architecture.
Sn8: Technical safety concept diagram
Sn9: Technical safety requirements summary
S6: Argument by safety analysis of safety concepts
Sn4: Risk summary with risk classification
C11: Safety concepts have been analyzed according via safety analysis methods to identify remaining risks. (e.g. unhandled malfunctions / faults / failures) .
S7: Argument by risk reduction measures
C12: Safety measures (Detection and/or prevention measures) for the identified risks have been defined.
Sn10: Risk prevention and detection measures
Sn11: Risk summary with residual risk classification
S8: Argument by development process according to identified hazards
C13: Safety tasks have been performed according to requirements of ISO26262 given the identified hazards.
Sn´2: Task overview
supported by
In Context of
G: Goal
S: Strategy
C: Context
Sn: Solution

Figure 16: Strategies of a safety case report modeled in GSN exploiting the SAFE MM

In the following sections we propose how these areas can be generated based on concepts of the SAFE Meta-model.

8.3.1 Area: Scope

The scope can be generated based on the items associated to the hazard and risk analysis. The following picture shows the relevant part of the SAFE Meta-model:

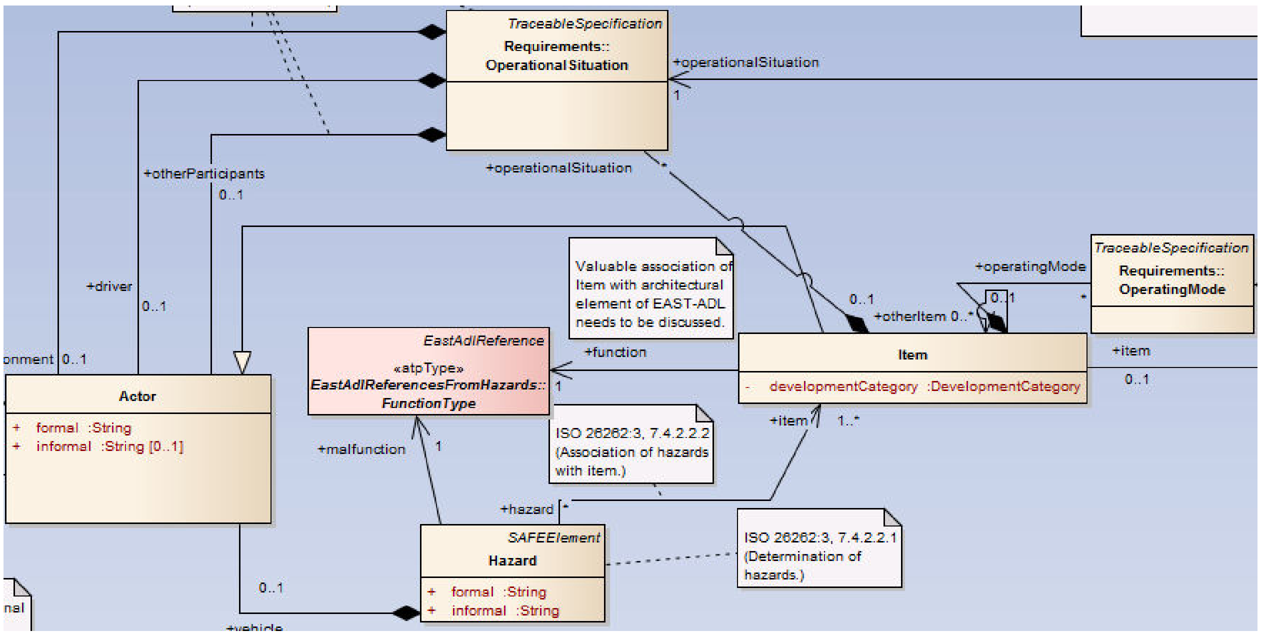

Figure 17: SAFE Meta Model Concepts for Scope

The following SAFE Meta-model concepts should be provided in the scope area of the safety case report:

- Requirements which support the scope of the safety case, e.g. in terms of unintended uses, misuses, limits or expected system life span
- Operating Modes
- Operational Situations

8.3.2 Area: System description

The system description should not provide full design detail but rather support the reader of the safety case report to make sense of the system hazards and requirements which are later described in the report.

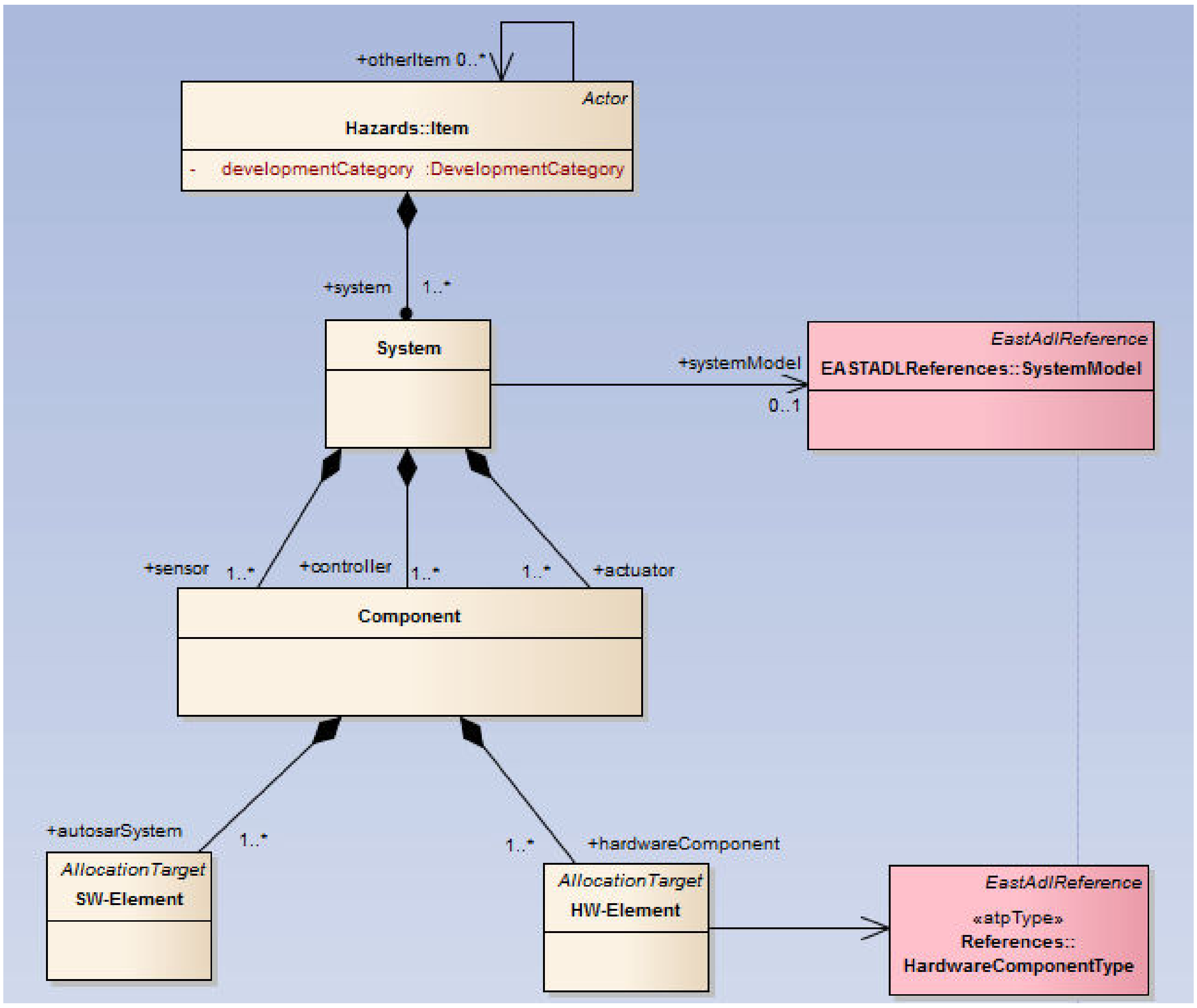

Figure 18: Item Architecture of the SAFE Meta-model

The following SAFE Meta-model concepts should be provided in the system description area of the safety case report:

- System descriptions
- Component descriptions
- System diagrams

8.3.3 Area: System Hazards

The system hazards area should list the key hazards posed by the system in order to summarize the identified hazards [5]. Hazards are an explicit part of the SAFE Meta-model as depicted in the following excerpt.

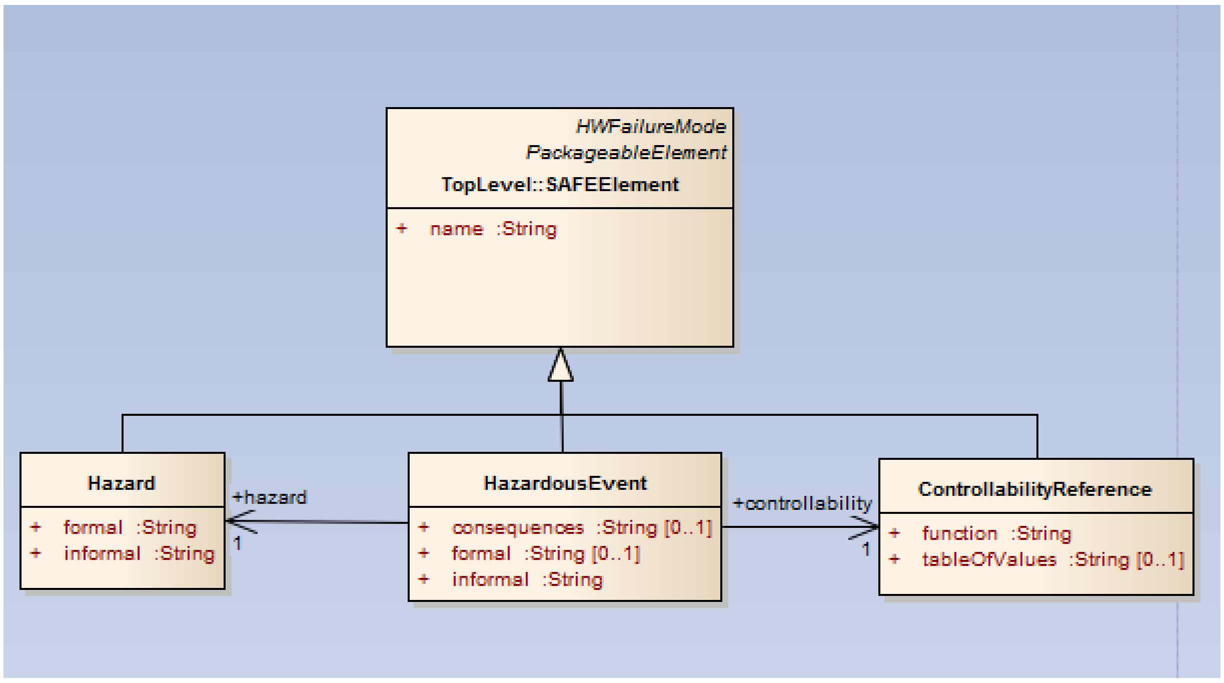

Figure 19: Hazards as defined in the SAFE Meta-model (Excerpt)

The following SAFE Meta-model concepts should be provided in the system description area of the safety case report:

- Hazard

8.3.4 Area: Safety Requirements

In the area safety requirements a number of sources for safety requirements must be taken into account [5] and are interpreted in the context of the SAFE Meta-model as outlined in the following points:

1. Safety requirements derived from hazard analysis (including safety goals as top level safety requirements)
2. Safety requirements which are the results of refinements from higher level safety requirements
3. Safety requirements which have been given directly by the customer or safety standards

The following picture shows the relevant excerpt of the SAFE Meta-model.

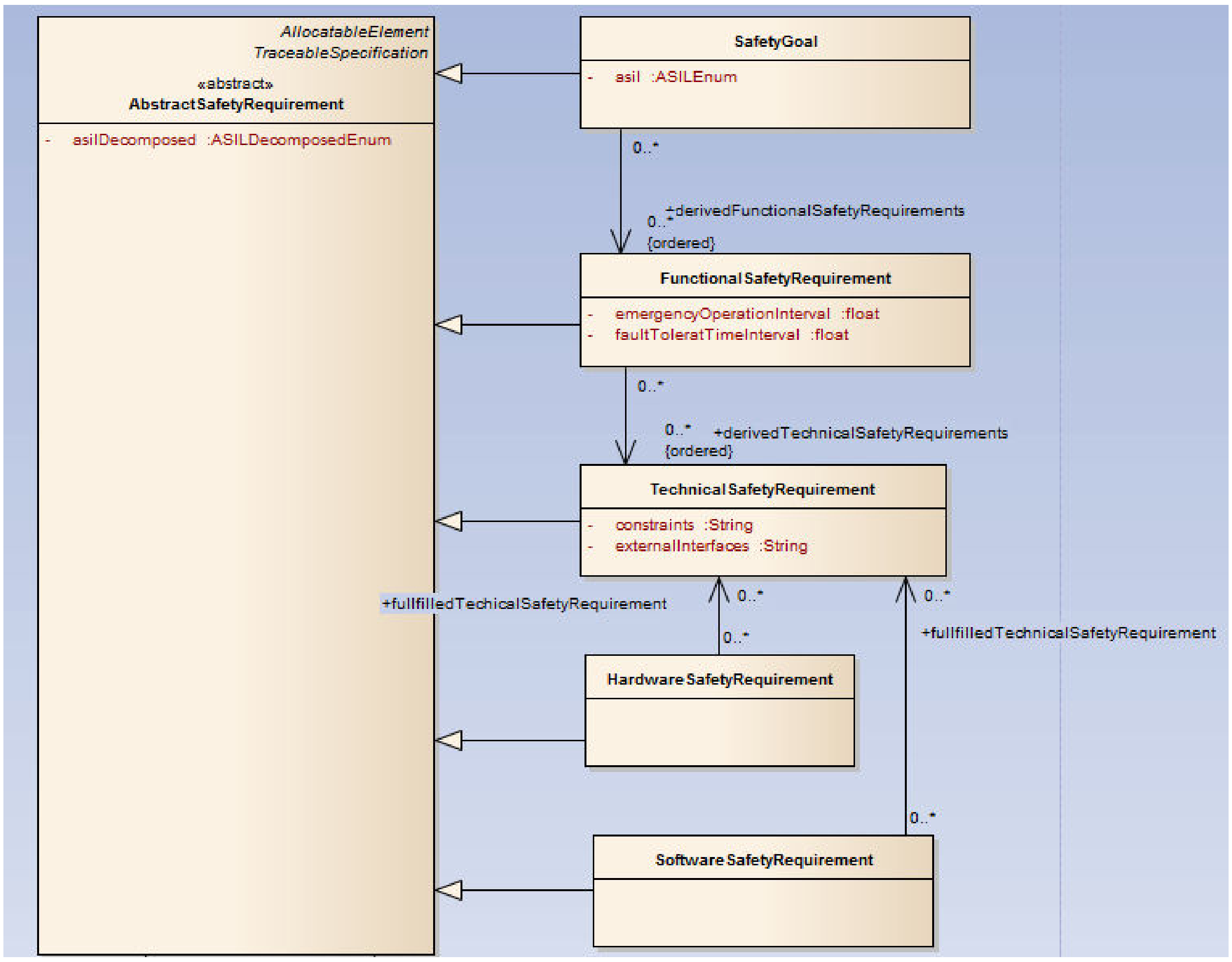
AllocatableElement
TraceableSpecification
«abstract»
AbstractSafetyRequirement
- asilDecomposed :ASILDecomposedEnum
SafetyGoal
- asil :ASILEnum
0..*
0..*
+derivedFunctionalSafetyRequirements
{ordered}
FunctionalSafetyRequirement
- emergencyOperationInterval :float
- faultToleratTimeInterval :float
0..*
0..*
+derivedTechnicalSafetyRequirements
{ordered}
TechnicalSafetyRequirement
- constraints :String
- externalInterfaces :String
+fullfilledTechicalSafetyRequirement
0..*
0..*
0..*
+fullfilledTechnicalSafetyRequirement
HardwareSafetyRequirement
0..*
SoftwareSafetyRequirement

Figure 20: Safety Requirement Expression as defined in the SAFE Meta-model (Excerpt)

The following SAFE Meta-model concepts should be provided in the system description area of the safety case report:

- Safety Goals
- Functional Safety Requirements
- Technical Safety Requirements
- Hardware Safety Requirements
- Software Safety Requirements

8.3.5 Area: Risk Assessment

The area risk assessment of the safety case report aims to describe the level of residual risk which is left after risk reduction measures have been applied [5].

The initial risk associated with a hazard is captured in the SAFE Meta-model in the hazardous event. However the residual risk after the implementation of risk reduction measures is not yet part of the SAFE Meta-model and can therefore not provided in the safety case report.

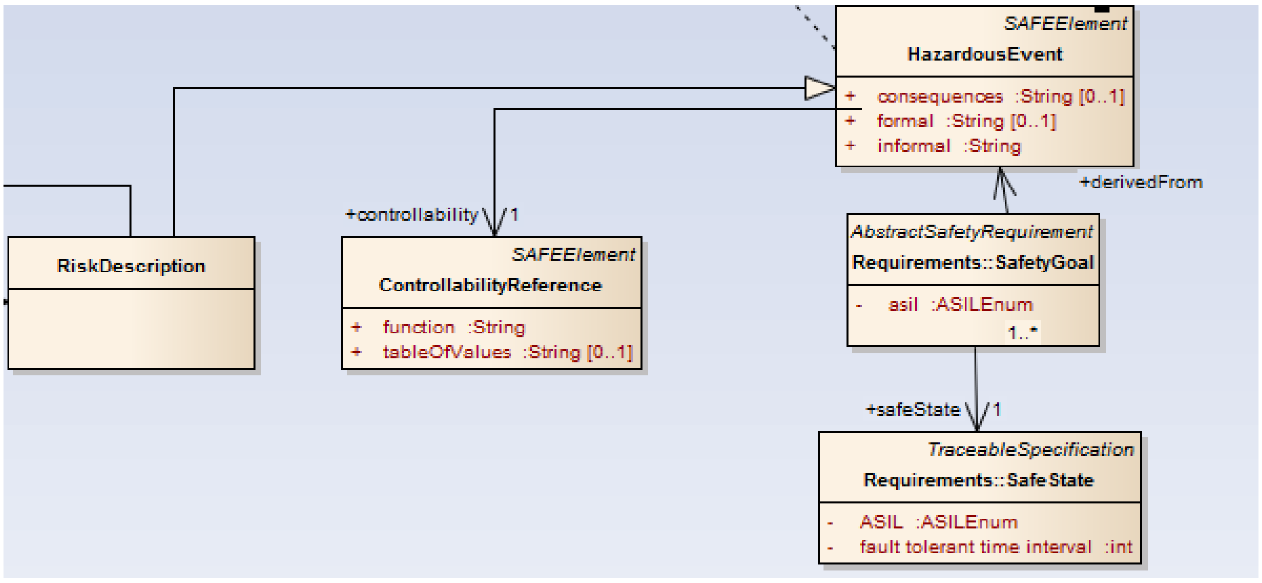

Figure 21: Initial risk description in the SAFE Meta-model

The following SAFE Meta-model concepts should be provided in the system description area of the safety case report:

- Hazardous Event

8.3.6 Area: Risk Reduction Measures

The area risk reduction measures describes means for reducing the probability of hazard occurrence or mitigation of hazard occurrence which have been integrated in the system design [5].

The following figures show excerpts of the SAFE Meta-model which could be exploited for this information.

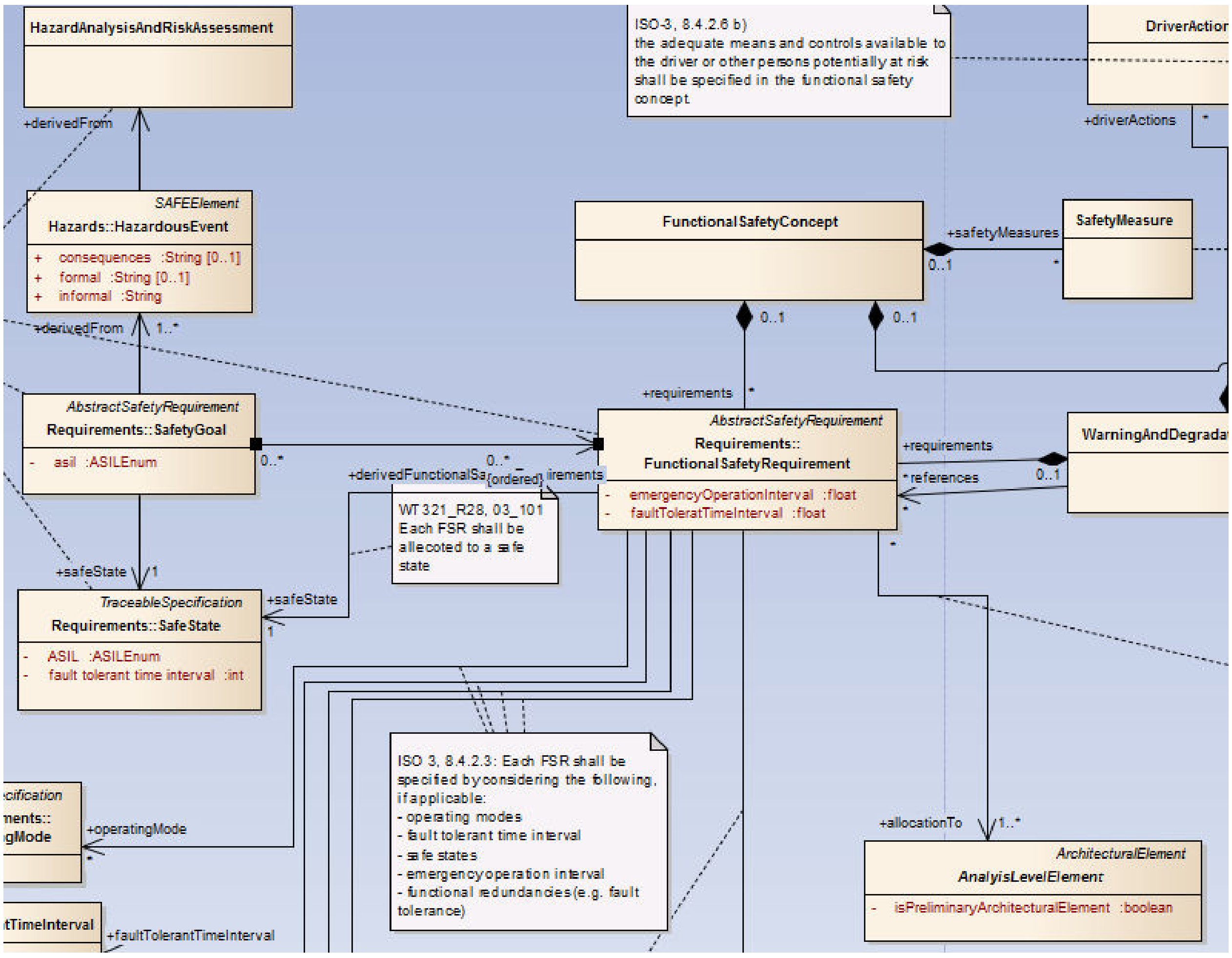
HazardAnalysisAndRiskAssessment
ISO-3, 8.4.2.6 b) the adequate means and controls available to the driver or other persons potentially at risk shall be specified in the functional safety concept
DriverActions
+derivedFrom
+driverActions
SAFEElement
Hazards::HazardousEvent
+ consequences :String [0..1]
+ formal :String [0..1]
+ informal :String
FunctionalSafetyConcept
+safetyMeasures
SafetyMeasure
+derivedFrom 1..*
+requirements
AbstractSafetyRequirement
Requirements::SafetyGoal
- asil :ASILEnum
AbstractSafetyRequirement
Requirements::FunctionalSafetyRequirement
- emergencyOperationInterval :float
- faultToleratTimeInterval :float
+requirements
+references
WarningAndDegradation
+derivedFunctionalSafetyRequirements {ordered}
WT 321_R28, 03_101 Each FSR shall be allecoted to a safe state
+safeState
TraceableSpecification
Requirements::SafeState
- ASIL :ASILEnum
- fault tolerant time interval :int
ISO 3, 8.4.2.3: Each FSR shall be specified by considering the following, if applicable:
- operating modes
- fault tolerant time interval
- safe states
- emergency operation interval
- functional redundancies (e.g. fault tolerance)
+operatingMode
+faultTolerantTimeInterval
+allocationTo 1..*
ArchitecturalElement
AnalysisLevelElement
- isPreliminaryArchitecturalElement :boolean

Figure 22: Functional Safety Concept of the SAFE Meta Model (Excerpt)

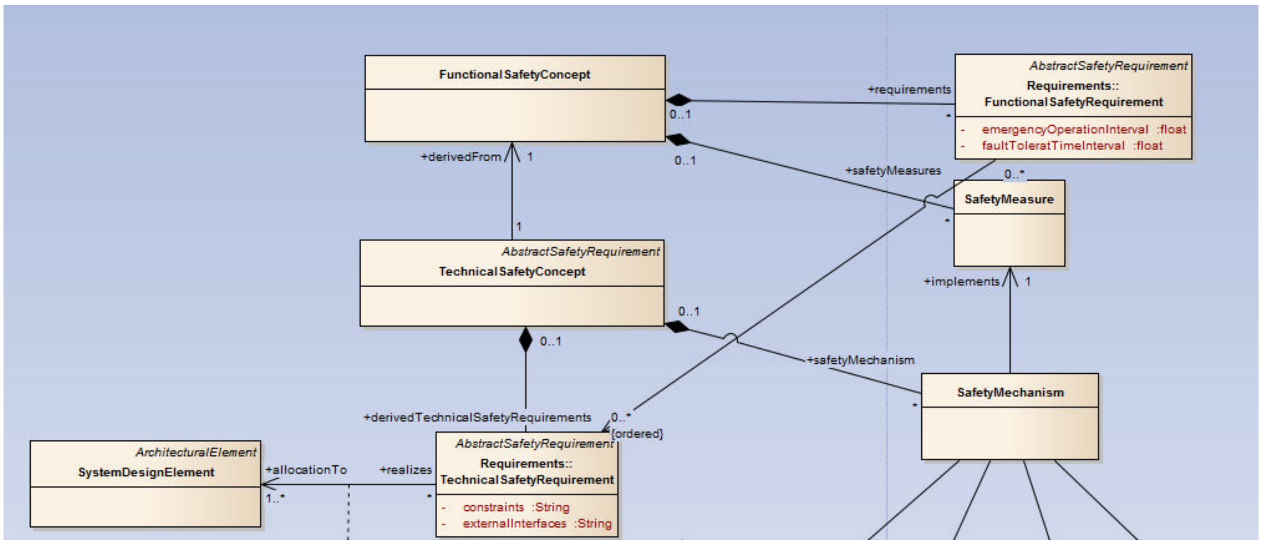
FunctionalSafetyConcept
+requirements
AbstractSafetyRequirement
Requirements::FunctionalSafetyRequirement
- emergencyOperationInterval :float
- faultToleratTimeInterval :float
+derivedFrom 1
+safetyMeasures
SafetyMeasure
AbstractSafetyRequirement
TechnicalSafetyConcept
+implements 1
+safetyMechanism
SafetyMechanism
+derivedTechnicalSafetyRequirements {ordered}
ArchitecturalElement
SystemDesignElement
+allocationTo
+realizes
AbstractSafetyRequirement
Requirements::TechnicalSafetyRequirement
- constraints :String
- externalInterfaces :String

Figure 23: Technical Safety Concept of the SAFE Meta Model (Excerpt)

The AnalysisLevelElement contributes to reducing or mitigating hazards by addressing the safety requirements which have been allocated to them.

The following SAFE Meta-model concepts should be provided in the risk reduction measures area of the safety case report:

- AnalysisLevelElements, which address one or more FunctionalSafetyRequirement, which have been created to reduce or mitigate risks from one or more Hazards
- SystemDesignElements, which address one or more TechnicalSafetyRequirements, which have been created to reduce or mitigate risks from one or more hazards

8.3.7 Area: Safety Analysis

The area safety analysis provides evidence that the risk reduction measures which have been reported in the previous area are sufficient [5]. Examples for means to achieve this are safety analysis methods (e.g. FMEA), inspections or in-service evidence. As in the other areas of the safety case report only a summary is required while details can be maintained in other documents.

The following figure shows an extract of the SAFE Meta-model that support analysis such as fault tree analysis and FMEA.

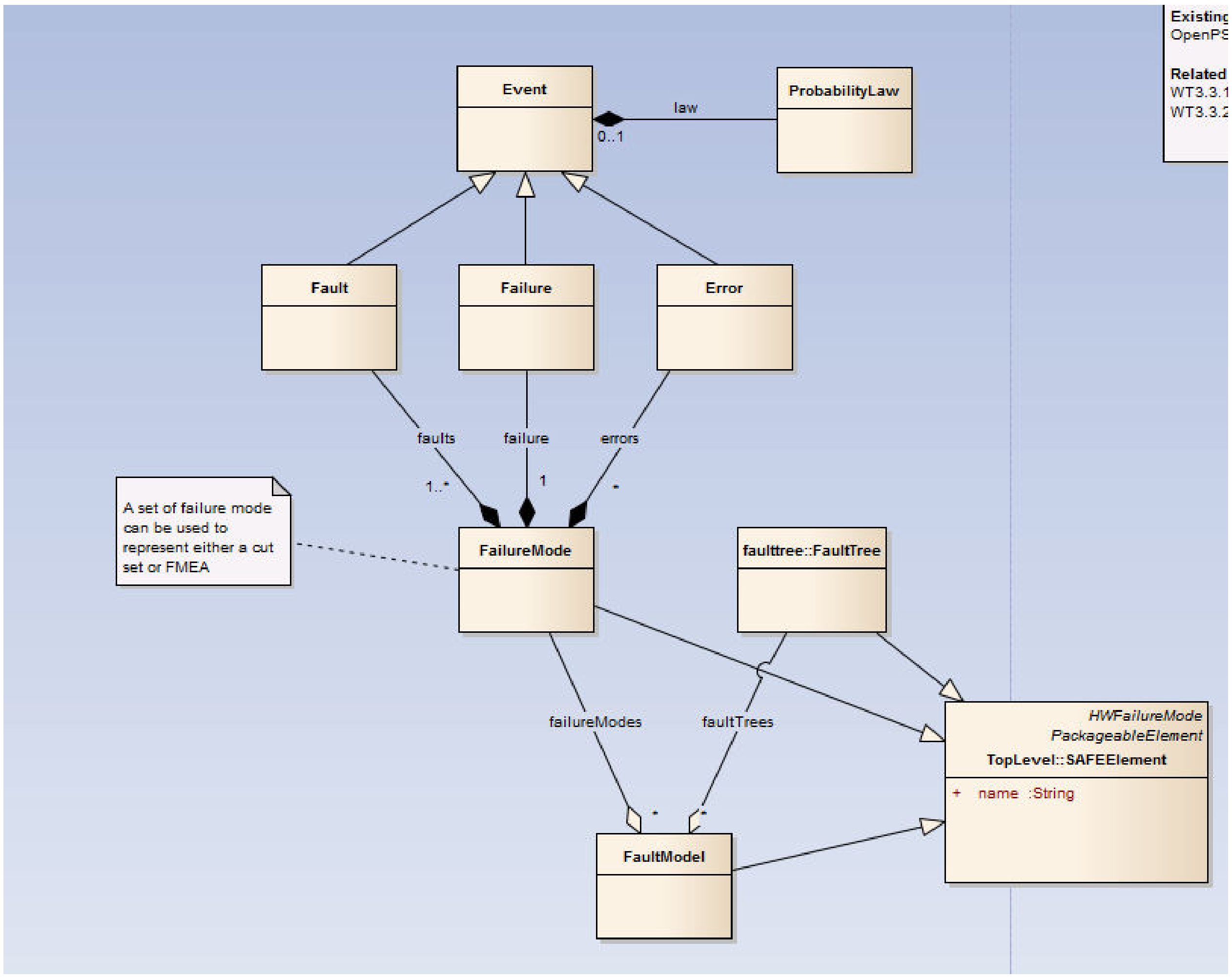

Figure 24: Concept for Fault Tree Analysis and Failure Modeling in the SAFE Meta Model (Excerpt)

A potential approach to document that the risk reduction measures defined previously are sufficient is to check if every failure which is identified in an FMEA can be associated with a risk reduction measure.

The following SAFE Meta-model concepts should be provided in the safety analysis area of the safety case report:

- Failures identified in an FMEA together with the associated risk reduction measures (e.g. architecture elements or requirements as prevention measures or test cases as detection measures)

However linking risk reduction measures with failures is not yet supported by the SAFE Meta-model. This is an obvious point for improvement of the SAFE Meta-model.

9 Interdependencies with other work tasks / packages

The relationships between WT 3.1.3 and other work task are shown in the following Figure 25.

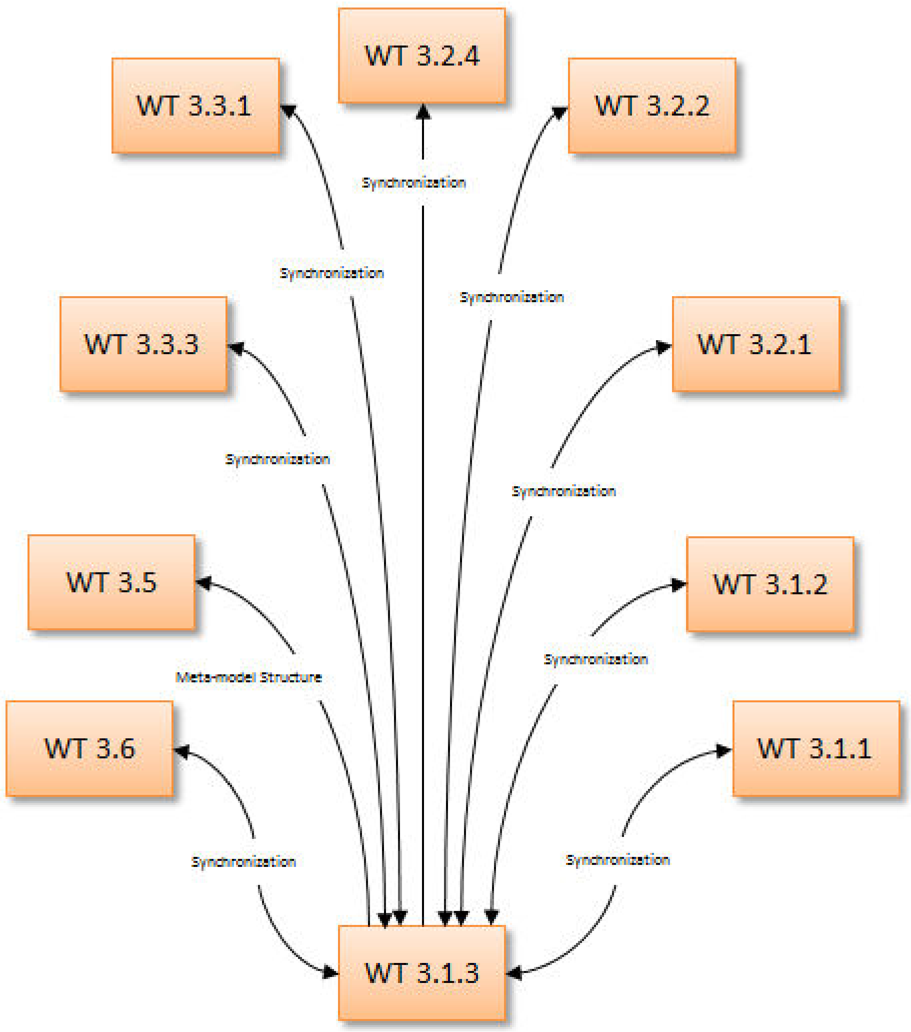

Figure 25: Relationships between WT 3.1.3 and other WTs

As it can be seen in the figure, there are various dependencies between WT 3.1.3 and other work tasks. In the following these interdependencies are described in more detail.

As shown in the figure, there is a relationship between WT 3.1.3 and WT 3.5 concerning the Meta-model structure. In particular this means that WT 3.1.3 has to contribute a respective Meta-model part for the integration in the SAFE Meta-model which is developed within WT 3.5. In order to ensure correctness of structure and settings as well as easy integration of the Meta-model parts into the combined SAFE model WT 3.5 set up a guideline document and a master document for the model so that each work task can work with replicas. By doing this, the consistency can be ensured.

Interdependency exists with the work task 3.2.1. Within this work task the Meta-model principles for the functional safety concept will be developed. Since the functional safety concept includes the functional safety requirements which are derived from the safety goals which are a starting point of the WT 3.1.3 – elements a strong communication is necessary to ensure that all artifacts needed for the derivation are present. Moreover, seamless traceability from the artifacts of the hazard analysis and risk assessment to the functional safety requirements needs to be established and can be supported by safety case expressions. Finally, safety concepts are prime candidates for the use of patterns as detailed in Section 6.3.

Due to the fact that not only a seamless traceability from safety goals to functional safety requirements has to be established but also that safety goals are top-level safety requirements and need to fulfill the same requirements on safety requirement expression given in the ISO 26262 as functional or technical safety requirements there is the need for synchronization between the work packages 3.1.1 and 3.1.2 and 3.1.3.

An additional synchronization link is established between WT 3.1.1 and WT 3.1.3. Within WT 3.1.3, the fundaments for expressing safety goals and hazardous events are established. Safety goals form the basis for any safety case and an important aspect for this documentation is the tracing from hazards to their solutions; therefore an exchange with respect to the representation of all included artifacts has to take place.

With respect to WT 3.2.4 dealing with the handling of COTS there is as well the need for synchronization. COTS components' hazards should already be known and could come with encapsulated safety cases for their inner workings at the very least, which could then be integrated into the safety case at the COTS inclusion point. Furthermore, these components have to be integrated in the hazard analysis and risk assessment and the safety case reassessed from a system level perspective.

Similarly WT3.6 introduces safety mechanisms, which represent tried and trusted solution for known and repeating problems. This is also a prime candidate for encapsulating safety case snippets with every mechanism, which can be imported and incorporated into the safety case at the safety mechanism insertion point.

Another bi-directional relationship exists between WT 3.1.1 and WT 3.3.1 (Failure and cut-sets analysis) as well as (3.3.3) due to the fact that analyses results represent inputs for the safety case.

Besides the relationships already mentioned there is the need for communication between WT 3.1.3, 3.1.1 and WT 3.2.2 concerning safety goals. There is the need for relating hardware failure information to safety goals in order to determine the role of violation of safety goals which then allows calculating hardware metrics, all of which flow into a safety case.

The degree of synchronization inclusion of the information performed is directly linked to the degree of automation achieved, and can vary from none to manual links to high, with cross class linkages.

10 Conclusions and Discussion

This document provides information about the safety case concept as adopted by many safety critical industries describes the proposed methodology for safety case modeling and documentation as well as for an extension of the SAFE Meta-model for hazard and environment modeling.

Besides giving an overview on the relevant parts of ISO 26262 the requirements arising from WT 2.1 (ISO 26262 Analysis), a focal part of this deliverable lies in the presentation of the methodology for safety case modeling and documentation. This methodology is compliant to the requirements given in ISO 26262 and in addition comprises aspects arising from experiences in the development of automotive systems as well as other safety critical industries (such as defense, railways and aerospace). However, the ability to describe and link development artifacts in a safety case relevant context is only half the story. How this capability is employed is the other half of this document. The concepts of solution and design patterns are introduced, along with the concept of compositional argumentation. These concepts are far from fully matured and therefore, the initial concepts presented in this paper can be seen as a basis for further development.

The initial contribution to the SAFE Meta-model presented in this deliverable provides the possibility to link safety-critical artifacts together to form an ISO 26262 compliant safety case. At the same time the requirements coming from the methodology are considered. In case the methodology is extended there might also arise the need to adapt the corresponding part of the SAFE Meta-model, including those developed in other work tasks.

Since it is an objective to reuse EAST-ADL as much as possible the status of the current version of EAST-ADL is presented and initial proposals for extensions are formulated. However, these proposals need to be further elaborated in future. For the proposed extension of the SAFE Meta-model EAST-ADL references are used whenever possible.

The specification described in this work will be further developed during the ongoing implementation phase and could be expanded to include new concepts. Most promising among these is the capability for compositional argumentation introduced in Section 6.3, as well as the concept of assured safety arguments, a new structure proposed by Hawkins and Kelly in [4] for introduces a confidence argument that documents the confidence in the structure and evidence of the safety argument.

11 Abbreviations used in D3.1.3

ASIL	Automotive Safety Integrity Level
ATESST	Advancing Traffic Efficiency and Safety through Software Technology
EPS	Electric Power Steering
GSN	Goal Structuring Notation
EAST-ADL	Electronic Architecture and Software Tools- Architecture Description Language
HA	Hazard Analysis
RA	Risk Analysis
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
HW	Hardware
SAFE	Safe Automotive soFtware architEcture
WT	Work Task

12 References

[1] International Organization for Standardization: ISO 26262 Road vehicles - Functional safety. (2011)

[2] S. Toulmin: “Uses of Argumentation”. Cambridge University Press, 1958; 2nd edn., 2003

[3] The ATESST Consortium: “ATESST Project Deliverable D2.2.2”, 2007

[4] R. Hawkins, T. Kelly, J. Knight and P. Graydon: “A New Approach to Creating Clear Safety Arguments”. Advances in Systems Safety - Proceedings of the Nineteenth Safety-Critical Systems Symposium, Southampton, UK, February 8-10, 2011. Springer 2011

[5] T. Kelly, Arguing Safety – A Systematic Approach to Managing Safety Cases, PhD Thesis, Department of Computer Science, The University of York, 1998.

[6] R. Weaver, The Safety of Software – Constructing and Assuring Arguments, PhD Thesis, Department of Computer Science, The University of York, 2003.

[7] F. Ye, Justifying the Use of COTS Components within Safety Critical Applications, PhD Thesis, Department of Computer Science, The University of York, 2005.

[8] R. Hawkins, K. Clegg, R. Alexander, T. Kelly, Using a Software Safety Argument Pattern Catalogue: Two Case Studies, in F. Flammini, S. Bologna, and V. Vittorini (Eds.): SAFECOMP 2011, LNCS 6894, pp. 185 - 198. Springer, Heidelberg, 2011.

[9] M. Jaffe, R. Busser, D. Daniels, H. Delseny, G. Romanski, Progress Report on Some Proposed Upgrades to the Conceptual Underpinnings of DO178B/ED-12B, In Proceedings of the 3rd IET International Conference on System Safety, 2008.

[10] T. Kelly, Concepts and principles of compositional safety case construction, Technical Report COMSA/2001/1/1, The University of York, 2001.

[11] R. Hawkins and T. Kelly, A Software Safety Argument Pattern Catalogue, Department of Computer Science, The University of York, 2008.

[12] GSN Community: “The Goal Structuring Notation GSN Community Standard Version 1”. Origin Consulting (York) Limited, 2011

[13] T. Kelly and R. Weaver: “The Goal Structuring Notation . A Safety Argument Notation”. Proc. DSN 2004 Workshop on Assurance Cases, 2004

[14] Kelly, Tim P., and John A. McDermid. "Safety case construction and reuse using patterns." 16th International Conference on Computer Safety, Reliability and Security (SAFECOMP 1997). 1997.

[15] C. Alexander. A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York, 1977.

[16] M. Khalil “Pattern-based methods for model-based safety-critical software architecture design” ZeMoSS 2013 Workshop at the SE 2013 in Aachen. February 2013.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. “Design Patterns: Elements of Reusable Object-Oriented Software.” Addison-Wesley, Boston, MA, USA, 1997.

[18] K. Beck and W. Cunningham. “Using pattern languages for object-oriented programs.” In Position Paper for the Specification and Design for Object-Oriented Programming Workshop, The 3rd Conference on Object- Oriented Programming Systems, Languages and Applications, Orlando, USA, 1987.

[19] J. O. Coplien. Advanced C++ programming styles and idioms. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

[20] J. O. Coplien. Idioms and patterns as architectural literature. IEEE Softw., 14(1):36–42, 1997.

[21] D. Manolescu, M. Voelter, and J. Noble. “Pattern Languages of Program Design 5.” Addison-Wesley Professional, 2005.

[22] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. “Pattern-oriented software architecture: a system of patterns.” John Wiley & Sons, Inc. New York, 1996.

[23] B. P. Douglass. “Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems.” Addison-Wesley, New York, Boston, MA, USA, 2002.

[24] R. Hanmer. “Patterns for Fault Tolerant Software.” Wiley Publishing, 2007.

[25] L. Rising. “Design patterns in communications software.” Cambridge University Press, New York, NY, USA, 2001.

[26] M. Pont. “Patterns for Time-Triggered Embedded Systems: Building reliable applications with the 8051 family of microcontrollers.” ACM Press, New York, 2001.

[27] M. J. Pont and M. P. Banner. “Designing embedded systems using patterns: a case study.” Journal of Systems and Software, 71(3):201–213, 2004.

[28] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann,and P. Sommerlad. „Security Patterns: Integrating Security and Systems Engineering.” John Wiley & Sons, March 2006.

[29] N. Yoshioka, H. Washizaki, and K. Maruyama. “A survey on security patterns.” Progress in Informatics:, 5:35–47, 2009.

[30] AutoFOCUS 3, research CASE tool, af3.fortiss.org, 2012 fortiss GmbH

[31] European Organisation for Civil Aviation Equipment, EUROCAE ED-79/ARP-4754: Certification considerations for highly integrated aircraft, 2011.

[32] A. Armoush, “Design Patterns for Safety-Critical Embedded Systems”, Ph.D. Thesis, RWTH-Aachen, 2010

[33] „Funktionale Sicherheit: Grundzüge sicherheitstechnischer Systeme“, Börcsök, J. / VDE-Verlag, 2011

[34] European Organization for Civil Aviation Equipment (EUROCAE) and United States Department of Defense (DOD), EUROCAE ED-12B/DO-178B: Software considerations in airborne systems and equipment certification, 1992.

[35] United States Department of Defense (DOD), DO-178C: Software considerations in airborne systems and equipment certification, 2012.

[36] European Committee for ElectroTechnical Standardization, EN-50126 Railway Standard, www.cenelec.eu, 2008.

[37] IEC 61508.

[38] Papadopoulos Y., McDermid J.A., The Potential for a Generic Approach to Certification of Safety-Critical Systems in the Transportation Sector, Reliability Engineering and System Safety, 63(1):47-66, Elsevier Science,1999.

[39] “Engineering a safer world”, Nancy Leveson / MIT Press, 2011

[40] „Funktionale Sicherheit von mechatronischen Systemen bei mobilen Arbeitsmaschinen“, Martinus, M. / Technische Universität München, 2004

[41] Stefan Wagner, Bernhard Schätz, Stefan Puchner, Peter Kock, A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models, Proc. International Symposium on Software Reliability Engineering (ISSRE '10), IEEE Computer Society, 2010

[42] Tim Kelly, A Systematic Approach to Safety Case Management, SAE 2004 World Congress & Exhibition, March 2004, Detroit, MI, USA, Session: Safety-Critical Systems.

[43] Weihang Wu and Tim Kelly. 2004. Safety Tactics for Software Architecture Design. In *Proceedings of the 28th Annual International Computer Software and Applications Conference - Volume 01* (COMPSAC '04), Vol. 1. IEEE Computer Society, Washington, DC, USA, 368-375.

[44] The EAST-ADL Consortium. EAST-ADL V2.1. http://www.east-adl.info/

[45] The SAFE Consortium. Deliverable D3.1.1a “Final proposal for extension of Meta-model for hazard and environment modeling”. ITEA2. 2013.

[46] The SAFE Consortium. Deliverable D3.5a “Initial proposal for Meta-model definition”. ITEA2. 2012

13 Acknowledgments

This document is based on the SAFE project in the framework of the ITEA2, EUREKA cluster programme Σ! 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019, and by the French Ministry of the Economy and Finance (DGCIS). The responsibility for the content rests with the authors.