UDC 004.056:004.77

K.I. LOBACHOVA

National Tavrida University, Simferopol, Ukraine

THE CONCEPT AND ARCHITECTURE OF SAFETY CASES: ELEMENTS OF ANALYSIS

The safety case and trust case concepts are presented, several development methodologies are discussed. The structures of the three most common notations for representing safety cases are reviewed, with their main elements being thoroughly described and the notation argument models schematically illustrated and analyzed. Different safety systems are compared in terms of the development approaches, presentation techniques and information sources used for safety cases implementation. The general scope, goals and main principles of ForSyDE Modelling Framework are introduced.

Key words: *Safety case, trust case, notations, Toulmin, ASCAD, GSN, ForSyDE.*

Introduction

In our modern world, there are many industrial processes that are of great social concern because of their complexity and the risk of potential failures which can endanger human lives or lead to serious environmental problems. For almost any system the most effective means of limiting liability (accident risk) is to implement an organized system safety function, beginning at the conceptual design phase and continuing through to its development, fabrication, testing, production, use and ultimately disposal.

In the present paper we outline a safety case methodology, which has been actively developing in recent years. Many scientists such as Peter Bishop, Tim Kelly, J Górski and others have been analyzing and improving it in their works ([1], [3], [6], [8]). We also examine different notations sketching the structure and the functions of their components, and describe the modern frameworks used in the area.

The paper is structured into five main sections: the first one introduces a safety case methodology; the second provides an overview of the most common notations; the next provides some suggestions for safety case implementation; the fourth chapter outlines the Trust Case methodology and describes Trust-IT framework for the trust case development; and the last one examines a ForSyDE modeling framework which can be considered an element of safety case methodology. The paper ends with some concluding remarks and statement of our future research plans.

1. Safety case concept overview

Software is increasingly used in many different applications. Of course, not all of the software products have the same criticality level. In fact, there are three classes of systems that differ in respect to safety. They are defined as: safety systems, safety-related systems and systems not important to safety. We are only interested in the first and second classes as they are sensitive to safety and require a certain work to be done to ensure safe operation.

To start working in this field we need to examine the Safety Case concept.

In classical theory safety case is defined as "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". [1]

To implement a safety case it's necessary to:

– make an explicit set of claims about the system;

– produce the supporting evidence;

– provide a set of safety arguments that link the claims to the evidence;

– make clear the assumptions and judgments underlying the arguments;

– allow different viewpoints and levels of detail.

There are also several alternative definitions, let's consider the following more extended definition provided in the U.K. Ministry of Defense Ship Safety Management System Handbook [2]

A safety case is a comprehensive and structured set of safety documentation which is aimed to ensure that the safety of a specific vessel or equipment can be demonstrated by reference to:

– safety arrangements and organization;

– safety analysis;

– compliance with the standards and best practice;

– acceptance tests;

– audits;
– inspections;
– feedback;
– provision made for safe use including emergency arrangements".

From these two definitions it is clear that the safety case is a document. We should also mention that in some standards the safety case is often used as a logical concept. But the common practice is not to distinguish between the safety case as a logical concept and the safety case as a physical document and use this term to cover both meanings.

It is also worth to mention that adequately or acceptably safe is not an undefined concept. It is usually quite clearly expressed as prescriptive requirements, development codes or assessment principles.

For example, Defense Standard 00-55 expresses requirements concerning the development and assessment of safety critical software systems and even goes so far as to define the expected structure and contents of safety case reports. Prescriptive requirements are a third party expression of a high-level safety argument – where meeting requirements implies some degree of safety. The safety case must clearly identify and address applicable requirements [3].

Whilst there are some variations between the recommendations of the standards the following list illustrates the most typical headings expected within a safety case report [3].

– Scope;
– System Description;
– System Hazards;
– Safety Requirements;
– Risk Assessment;
– Hazard Control / Risk Reduction Measures;
– Safety Analysis / Test;
– Safety Management System;
– Development Process Justification;
– Conclusions.

2. Safety case notations

The first conceptual work in the area was conducted by Toulmin in the 1950s. He developed a framework and graphical notation for representing the structure of an argument. Toulmin's ideas have influenced the most important methodologies for safety case development.

Different safety case notations such as Claims-Argument-Evidence and Goal Structuring Notation have been developed based on the Toulmin's work.

Below we consider each notation more closely.

2.1. Toulmin's notation

Toulmin's notation describes the scheme for the structure of a typical argument. The model is presented in the fig. 1.

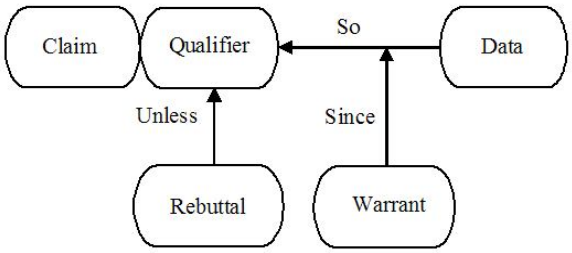

Fig. 1. Toulmin's argument model

The scheme above contains the following components:

– Claim – a certain conclusion to be demonstrated, some statements about the property of the system or its subsystem which is under consideration (a requirement, a property that the system should possess and so forth);
– Data – the facts appealed to as a foundation for the claim;
– Warrant – the reason why the evidence supplied should be accepted, the warrant that the facts indeed support the claim. It actually links the data and other grounds to the claim;
– Qualifier – the degree of confidence that can be placed on the claim;
– Rebuttal – counter-arguments that can be used, or certain conditions under which the claim may be falsified.

Validity and soundness of such arguments can be considered.

The validity of an argument is the acceptability of the justification (i.e. the warrant) used. The premises (i.e. the data) should be relevant (they should influence the conclusion, i.e. the claim) and adequate (truthfulness of the premises should imply truthfulness of the conclusion).

The soundness of an argument can be understood as the validity of the inference and the acceptability of the premises [4].

Following Toulmin's approach, more recent notations with supporting methodologies have been developed. And the following section describes a famous Claims-Argument-Evidence or ASCAD notation.

2.2. ASCAD notation

ASCAD stands for "Adelard Safety Case Development". ASCAD notation is a notation that uses a "claims-arguments-evidence" motif for representing

argument structure [1]. The notation scheme is provided in the fig. 2.

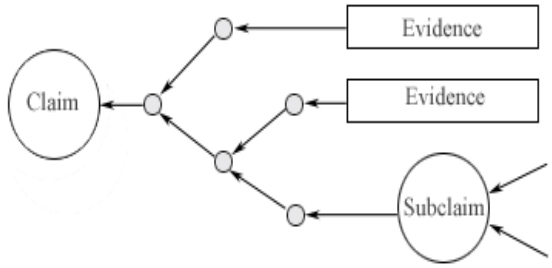

Fig. 2. ASCAD argument model

The main elements of the structure are:

– Claim about a property of the system or some subsystem.

– Evidence which is used as the basis of the safety argument. This can be facts, (e.g. based on established scientific principles and prior research), assumptions, or subclaims, derived from a lower-level sub-argument.

– Argument linking the evidence to the claim, which can be deterministic, probabilistic or qualitative.

– Inference the mechanism that provides the transformational rules for the argument.

It is also possible to have two (or more) independent arguments supporting the same claim.

It is worth to mention that "evidence" can be a sub-claim produced by a subsidiary safety-case. In other words, claims can take a set of sub-claims as their grounds and we can demonstrate the top level claim by showing that the lower level grounds are justified and making sure all the arguments are valid.

This means that there can be a relatively simple top-level argument, supported by a hierarchy of subsidiary safety cases. This structuring makes it easier to understand the main arguments and to partition the safety case activities.

2.3. GSN notation

GSN or Goal Structuring Notation is a graphics-based notation that explicitly represents the individual elements of any safety argument (requirements, claims, evidence and context) and (perhaps more significantly) the relationships that exist between these elements (i.e. how individual requirements are supported by specific claims, how claims are supported by evidence and the assumed context that is defined for the argument).

The principal symbols of the notation are shown in the fig. 3 (with example instances of each concept).

Below the primary elements of the notation are explained [5]:

– A goal states a claim (proposition or statement) that is to be established by an argument. A GSN diagram (called a goal structure) will usually have a top-level goal, which will often be decomposed into more goals.

– A strategy describes the method used to decompose a goal into additional goals.

– A solution describes the evidence that a goal has been met.

– The context associated with another GSN element lists information that is relevant to that element. For example, the context of a particular goal might provide definitions necessary to understand the meaning of the goal.

– An assumption is a statement that is taken to be true, without further argument or explanation.

– A justification explains why a solution provides sufficient evidence to satisfy a goal.

To construct an argument, the elements of the GSN notation are linked together using directed lines. An example of a safety argument goal structure is shown in the fig. 4.

The key benefit of GSN is that it improves the comprehension of the safety argument amongst all of the key project stakeholders – it is the reason why GSN has been adopted by a growing number of companies within safety-critical industries (such as aerospace, railways and defense) for the presentation of safety arguments within safety cases.

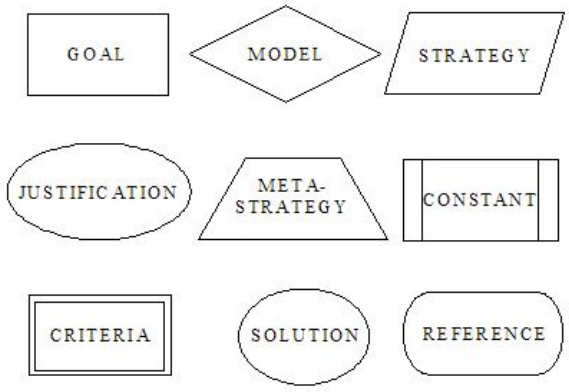

Fig. 3. GSN notation elements

3. Elements of technique

In the previous chaptes we described the general concept and structure of safety cases. Now let us offer some suggestions for producing a safety case. The approach will depend on a system you are working on and will be different for different (simple and complex, safety critical and safety-related) systems. Some basic-things that should be taken into account are provided in the table below (tabl. 1).

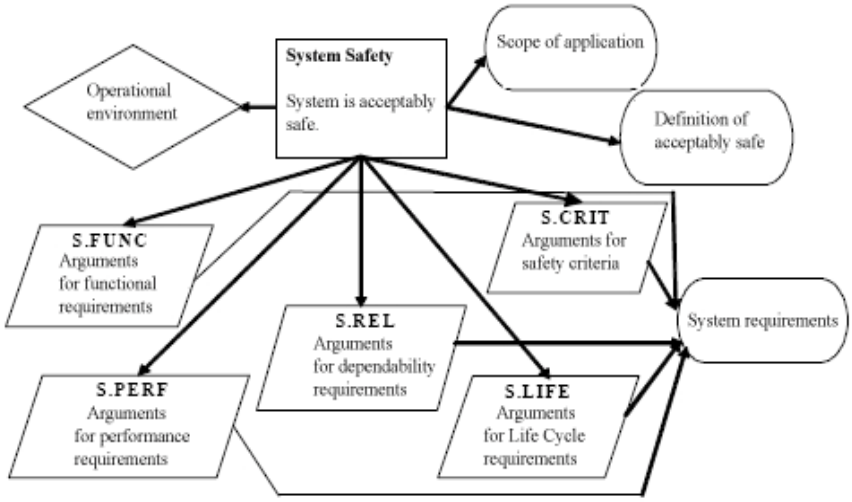

Fig. 4. A sample Goal Structure

4. Trust case

Now if we look at the issue on a broader scale, we can say that there are argument structures that are used to demonstrate properties different from safety. Such structures are called trust cases or assurance cases. A trust case is a documented base that provides satisfactory (from a given viewpoint) justification for a specified set of claims (regarding properties of an object considered for a given purpose in a given environment) to make a judgment about their trustworthiness.

This definition was provided by Górski [6], pro-

Table 1

Implementing safety cases for different systems

Safety case / System type	Hierarchy	Simplicity	Notations	Information
Complex safety-critical systems	+ Layered safety case with top-level and sub-level claims. Relatively simple top-level arguments are supported by a hierarchy of subsidiary safety cases.	+ Simplicity is essential for this type of systems. Use simple claims, clear supporting evidence.	Use advanced preferably graphical notations that explicitly represent the individual elements and the relationships between them. For example, GSN notation.	Use only reliable internal source of information. The data must be accurate, precise, timely identified and complete.
Simple safety-critical systems	– As system is simple, subsidiary safety-cases are not needed.	+ Simplicity is required.	Use any notation with simple element structure.	Internal source of accurate information is preferred.
Complex safety-related systems	+ Layered safety case with subsidiary safety cases created for subsystems.	– Simplicity is preferred but not required.	GSN or ASCAD notation is recommended.	Internal testing information can be used together with adequate operational feedback.
Simple safety-related systems	– Structuring is not needed.	– Simplicity is not necessary.	Use any notation suitable for representing the case.	Functional testing and adequate operational feedback can be used to compensate the lack of internal information.

fessor at Gdansk University of Technology. The viewpoints mentioned in the definition represent concerns of viewers of the trust case (stakeholders, auditors, etc.) about the object (a system, organization, etc.) under consideration. The documented base mentioned can include any evidence and justification, and is represented as an argument structure.

Górski has also originated the Trust-IT methodology for safety case development based on the Toulmin's notation we examined above (fig. 1). It is one of the most advanced methodologies to the date, with a Trust-IT Framework tool created for the development of trust cases and their application in different scenarios.

The Trust-IT framework consists of three components:

– Application component – explains possible usage scenarios of trust cases.

– Methodological component – explains how to develop and maintain trust cases, defines the language for trust case development, the syntax, semantics and typical argument patterns for trust cases, as well as business processes and procedures related to the application scenarios.

– Tool component – provides support for full-scale exploitation of the two other components.

The structure of the Trust-IT argument model is presented on the fig. 5.

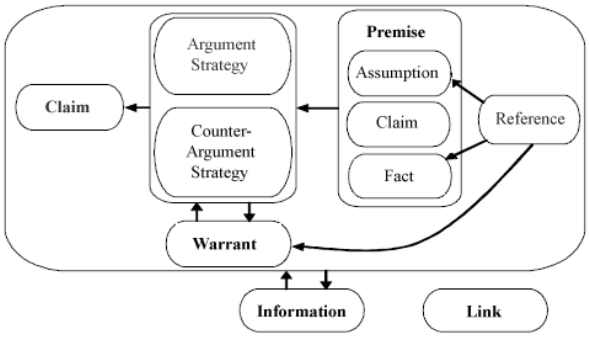

Fig. 5. Trust-IT argument model

A brief description of the nodes based on the [4] is provided blow:

– Claim – a proposition which expresses a desired property; each claim requires further justification, it is supplemented by an explicit argument.

– Argument Strategy – the basic idea how to demonstrate the conclusion and what the criteria for the selection of the premises are. A claim can have more than one argument strategy, in which case they provide independent arguments of the conclusion.

– Counter-Argument Strategy – the basic idea on which rebuttal of a supported claim is based. They can be treated as argument strategies for the negation of the conclusion. A claim can have multiple counter-argument strategies. It can also have argument strategies together with counter-argument strategies.

– Warrant – a conveyance relationship between premises and a conclusion. It explains why in given circumstances it necessitates or makes liable obtaining the conclusion if the premises obtain.

– Assumption – a premise which is taken as it is, without any further justification. By an assumption a property which is not dependent on the party which provides the trust case is represented.

– Fact – a statement or assertion of verified information about something that is the case or has happened. It can be obvious information or information based on other sources, external to the trust case.

– Reference – a link to the external, with respect to a given trust case, world. It can point to any identifiable, external object, being usually, in practice, an entity pointed to by a URL address. By means of references objects which contain evidence related to the argument kept in the trust case can be integrated.

– Information – additional information, which is not part of the argument itself. It can be put anywhere in the trust case and contains explanatory information, which can help to understand the meaning of the trust case or it helps organize the trust case structure.

– Links – an internal pointer pointing from one element of the trust case at another. By using links •one can overcome the tree-like structure of the trust case and make it a directed acyclic graph.

Claims and warrants can be demonstrated using other claims that means that the structure can grow recursively. Therefore, trust cases can be developed by provision of more detailed arguments for claims and some of warrants until all of them are fully justified.

The argument model is very expressive and can be applied to represent both formal inference and informal arguments. It provides means of representing arguments based on highly formalized analysis as well as uncertain evidence and inductive inference, which are so common in real life situations.

5. ForSyDE modelling framework

ForSyDe stands for Formal System Design. It is a methodology developed by KTH (Royal Institute of Technology, Stockholm) with the objective to move system design (i.e. System on Chip, Hardware and Software systems) to a higher level of abstraction and to bridge the abstraction gap by transformational design refinement.

Further description is based on the official ForSyDe documentation [7]. ForSyDe is based on carefully selected formal foundations. First, the designer must supply a initial specification model, which can mix dif-

ferent *Models of Computation* (MoCs) i.e. Synchronous MoC, Untimed MoC etc.

It is modeled by a network of *processes* interconnected by *signals*. Each process is created by a *process constructor*, allowing separate communication and computation.

Then, the abstract specification model is refined by different design transformations into a detailed *implementation model*, which is finally translated into a target implementation language.

ForSyDe research currently pursues two main goals:

– System modeling with heterogeneous models of computation. Libraries for different computational models allowing the simulation of heterogeneous systems have been developed. The libraries stretch from continuous time to synchronous time models. Using ForSyDe's shallow-embedded DSL (Domain Specific Language) electronic systems with analog and digital parts can be simulated.

– Development of a transformational design refinement methodology. A methodology for transformational design refinement have been outlined. The system, initially described as an abstract specification model, is refined into a more detailed implementation through semantic preserving and non-semantic preserving design transformations. Then, the resulting implementation model is translated into a target language.

ForSyDe systems are modeled as networks of processes interconnected by signals. Processes are pure functions on signals, i.e. for a given set of input signals a process always gets the same set of output signals.

$$p : \underbrace{S \times S \times ... \times S}_{n} \rightarrow \underbrace{S \times S \times ... \times S}_{m}$$

They can also be viewed as a black box which performs computations over its input signals and forward the results to adjacent processes through output signals (fig. 6).

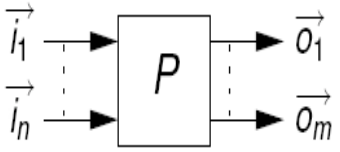

Fig. 6. Processes viewed as boxes

A process does not necessarily react identically to the same event applied at different times. But it will produce the same, possibly infinite, output signals when confronted with identical, possibly infinite, input signals provided it starts with the same initial state:

$$i_0 = i'_0, i_1 = i'_1, ..., i_n = i'_n \Rightarrow p(i_1, i_2, ..., i_n) = p(i'_0, i'_1, ..., i'_n)$$

Another key element in ForSyDe is Process Constructor. A process constructor PC takes zero or more functions f_1, f_2, ... , f_n (which determine behavior of the process) and zero or more values v_1, v_2, ... , v_n (determine the process configuration parameter or initial state) as arguments and returns a process $p \in P$:

$$p = pc(f_1, f_2, ..., f_n, v_1, v_2, ..., v_n)$$

The functions operate over the values carried by signals, not over signals themselves.

The structure is presented below (fig.7).

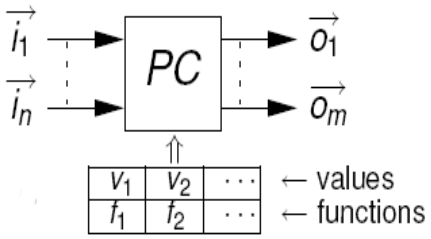

Fig. 7. A process constructor

The methodology defines a set of well-defined process constructors, which are used to create processes.

ForSyDe is implemented as a EDSL (Embedded Domain Specific Language) on top of the Haskell programming language with the implementation relying on different Haskell-extensions.

Conclusion

In this paper we discussed some elements of formal systems and tools as part of a Safety Case methodology, examined different notations and analyzed their structure, reviewed the frameworks and methodologies used in the area. All these information will be further used for preparing information technology and tool for assessing quality of critical software systems.

References

1. Bishop P. A Methodology for Safety Case Development / P. Bishop, 1998.

2. U.K. Ministry of Defense, "JSP 430 - Ship Safety Management System Handbook," Ministry of Defence, 1996.

3. Kelly T. A Systematic Approach to Safety Case Management / T. Kelly, 2003.

4. Cyra Ł. A Method of Trust Case Templates to Support Standards Conformity Achievement and Assessment / Ł. Cyra, 2008.

5. Holloway C.M. Safety Case Notations: Alternatives for the Non-Graphically Inclined? / C.M. Holloway, 2008.

6. Górski J. Trust Case – a case for trustworthiness of IT infrastructures / J. Górski // Cyberspace Security and Defense: Research Issues, 2005.

7. Formal System Design Tutorial [Електронний ресурс] – Режим доступу. http://www.ict.kth.se/forsyde/files/tutorial/tutorial.html.

8. Kelly T. Arguing Safety – A systematic Approach to Managing Safety Cases / T. Kelly, 1998.

9. Acosta. A Rising the abstraction level in System Design / A. Acosta, 2008.

10. Sander I. Modelling Adaptive Systems in ForSyDe / I. Sander, A. Jantsch, 2008.

11. Sklyar V.V. Software Quality Assessment and Expertise / V.V. Sklyar, V.S. Kharchenko (ed), 2008.

Поступила в редакцию 11.02.2009

Рецензент: д-р техн. наук, проф. Б.М. Конорев, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

КОНЦЕПЦИЯ И АРХИТЕКТУРА ОТЧЕТОВ ПО БЕЗПАСНОСТИ: ЭЛЕМЕНТЫ АНАЛИЗА

Е.И. Лобачева

Представлены концепции отчетов по безопасности и доверию, описаны методологии по их построению. Рассмотрены структуры трех наиболее распространенных нотаций для представления отчетов по безопасности, с подробным описанием основных элементов и схематичным представлением и анализом моделей аргументов этих нотаций. Проведено сравнение различных систем в отношении подходов к разработке, техник представления и информационных источников, используемых для реализации отчетов по безопасности. Указаны основные сферы применения, цели и принципы работы ForSyDE Modelling Framework.

Ключевые слова: Safety case, trust case, нотация, Тулмин, ASCAD, GSN, ForSyDE.

КОНЦЕПЦІЯ ТА АРХІТЕКТУРА ЗВІТІВ ПРО БЕЗПЕКУ: ЕЛЕМЕНТИ АНАЛІЗУ

К.І. Лобачова

Розглянуті концепції звітів про безпеку та довіру, описані методології їх побудови. Приведені та проаналізовані структури трьох найпоширеніших нотацій для створення звітів про безпеку, із детальним описом основних елементів та схематичним зображенням і аналізом моделей аргументів цих нотацій. Порівняні різні системи у відношенні підхіду до розробки, техніки зображення та джерел інформації, що використовуються при реалізації звітів про безпеку. Вказані основні сфери застосування, мети та принципи роботи ForSyDE Modelling Framework.

Ключові слова: Safety case, trust case, нотація, Тулмін, ASCAD, GSN, ForSyDE.

Лобачева Екатерина Игоревна – аспирант кафедры компьютерной инженерии, Таврический национальный университет, Симферополь, Украина, e-mail: kate.simferopol@gmail.com.