Developing Safe Control Systems using Patterns for Assurance

André Alexandersen Hauge
Institute for energy technology, Halden, Norway
University of Oslo, Norway
andre.hauge@hrp.no

Ketil Stølen
SINTEF ICT, Oslo, Norway
University of Oslo, Norway
ketil.stolen@sintef.no

***Abstract*—The Safe Control Systems (SaCS) method is a pattern-based method supporting the development of conceptual designs for safety critical systems. A pattern language offers support for the method by six different kinds of basic patterns, operators for combining patterns, and a graphical notation for visualising a pattern composition. Intended users of SaCS are system developers, safety engineers and HW/SW engineers. The method has so far been applied in two cases within different industrial domains. This paper demonstrates and presents experiences from the application of SaCS within the railway domain. We consider an interlocking system that controls the appliances of a railway station. We argue that SaCS effectively supports the establishment of requirements, a design satisfying the requirements, and an outline of a safety demonstration for the design.**

Keywords-conceptual design; pattern language; development processes; safety;

I. INTRODUCTION

This paper demonstrates and presents experiences from the use of a pattern-based method called Safe Control Systems (SaCS) to develop a conceptual design for a railway interlocking system.

SaCS has previously been tested out in the nuclear domain for the development of a reactor control system design [1].

The six kinds of basic patterns offered by SaCS are categorised according to two development perspectives: *Process Assurance*; and *Product Assurance*. Both perspectives detail patterns according to three aspects: *Requirement*; *Solution*; and *Safety Case*. We distinguish between basic and composite patterns. Each basic pattern contains an instantiation rule that may be used to assess whether it is correctly instantiated.

The basic SaCS patterns captures design solutions as well as commonly accepted safety engineering practices, e.g., development processes and activities, risk assessment methods, and other methods for providing safety assurance as reflected in international safety standards and guidelines, e.g., [2], [3], [4], [5]. Basic SaCS patterns are defined in a format inspired by classical literature on patterns [6], [7], [8]. It differs with respect to its explicit definition of inputs, outputs, and the instantiation rules that defines the transition from input to output for each pattern. The explicitly defined parameters facilitate easy combination of several patterns. The instantiation rules facilitate validation of the result of pattern instantiation based on the pattern definition and the given inputs. The composite patterns are expressed graphically and its notation is inspired by languages for system modelling, e.g., the modelling of patterns by collaborations, or modularisation of a specification by decomposition in UML [9], and literature related to visualisation, e.g., related to risk analysis [10] and general literature on visualisation of complex data [11].

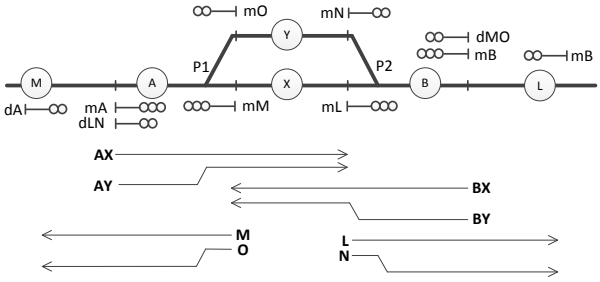

Figure 1. Railway Station Overview

The remainder of this article is structured as follows: Section II outlines the railway case on design of an interlocking system. Section III gives a short background on the SaCS method and its supporting pattern language. Section IV presents our hypothesis and main prediction. Section V to Section VIII exemplifies the stepwise application of the SaCS method and supporting pattern language in an example-driven manner for establishing a conceptual design for the railway interlocking system. Section IX outlines the results of applying SaCS. Section X presents related work, while Section XI concludes.

II. THE SYSTEM: RAILWAY INTERLOCKING

Fig. 1 illustrates the main appliances of a train station with two tracks.

The station in Fig. 1 is connected in both ends of the station area to neighbouring stations with a single track. An interlocking system controls the appliances associated with the station in order to safely control the movements of trains along defined train routes. The train station that is used as a case has a level crossing (Note: the level crossing is not depicted). The annotations in Fig. 1 denote the following:

- The circles with a letter inside, i.e., M, A, X, Y, B, and L denote the different track sections.

 ISBN: 978-1-61208-268-4

- All lines that end with two or three adjacent circles represent either a distant light signal, i.e., dA, dB, dLN, and dMO or a main light signal, i.e., mA, mB, mL, mM, mN, and mO.
- There are two points for switching traffic onto different tracks, these are identified as P1 and P2.
- The arrows illustrate the eight train routes, i.e., AX, AY, BX, BY, L, M, N, and O that are possible with the depicted track configuration.

The SaCS method was applied to develop a conceptual design for an interlocking system to control the appliances of a railway station with two tracks and a level crossing such that trains may move safely according to defined train routes. The interlocking system is one of many sub-systems, though the most critical to safety, in an overall system for controlling train movements. The conceptual design is intended to model a replacement of an existing system governing the interlocking rules only.

III. Background – SaCS

A. The SaCS Method

The method interleaves three main activities, each of which is divided into sub-activities:

S *Pattern Selection* – The purpose of this activity is to support the conception of a design by selecting: a) SaCS patterns for requirement elicitation; b) SaCS patterns for establishing design basis; c) SaCS patterns for establishing safety case.

C *Pattern Composition* – The purpose of this activity is to specify the use of the selected patterns by specifying: a) compositions of patterns; and b) instantiations of patterns.

I *Pattern Instantiation* – The purpose of this activity is to instantiate the composite pattern specification by: a) selecting pattern instantiation order; and b) conducting stepwise instantiation.

Pattern selection is supported by a selection map that is introduced in Section V-A. Pattern composition is supported by a pattern language outlined in Section III-B. Pattern instantiation is supported by instantiation rules defined for every basic pattern (defined in [12]).

B. The SaCS Pattern Language

Fig. 2 presents the main graphical elements used to illustrate the kind of patterns involved in a composite pattern.

The SaCS Pattern Language (SaCS PL) consists of patterns (basic SaCS patterns) of different types, and annotations for specifying how patterns are combined and applied in order to derive a conceptual design.

Every basic SaCS pattern is defined such that it may be used stand-alone. Every pattern is also defined such that it is easy to use several patterns together as every input and output parameter of a pattern is explicitly detailed.

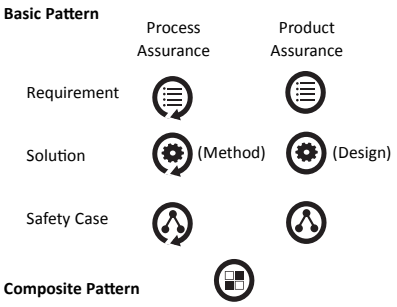

Figure 2. Icons for Visually Representing a Pattern

A composition of patterns (composite for short) may be expressed by, e.g., mapping an output parameter of a pattern to an input parameter of a second pattern and thereby defining a relationship between the patterns.

The different icons in Fig. 2 for representing a pattern reflect the different types of patterns in SaCS PL. Operators combine patterns. The operators will be introduced and explained by the examples provided in Section V to Section IX. A composite pattern may contain any type of SaCS pattern, i.e., basic patterns, composite patterns, or a combination of composite and basic patterns.

IV. Hypothesis

Success is evaluated based on the satisfaction of predictions. The hypothesis (H) and predictions (P) for the application of SaCS is defined below.

H: The SaCS method facilitates effective and efficient development of conceptual designs that are: 1) consistent; 2) complete; 3) correct; 4) comprehensible; 5) reusable; and 6) implementable.

Definition *A conceptual design is a triple consisting of a specification of requirements, a specification of design, and a specification of a safety case. The design characterises a system that satisfies the requirements. The safety case characterises a strategy for demonstrating that the design is safe with respect to safety requirements.*

We deduce the following prediction from the hypothesis with respect to the application of SaCS on the case described in Section II:

P: Application of the SaCS method on the railway case described in Section II results in a conceptual design that uniquely characterises the railway case and is easily instantiated from a composite SaCS pattern. Furthermore, the conceptual design is 1) consistent; 2) complete; 3) correct; 4) comprehensible; 5) reusable; 6) implementable.

Definition *A conceptual design instantiates a SaCS composite pattern if each element of the triple can be instantiated from the SaCS composite pattern according to the instantiation rules of the individual patterns and according to the rules for composition.*

 ISBN: 978-1-61208-268-4

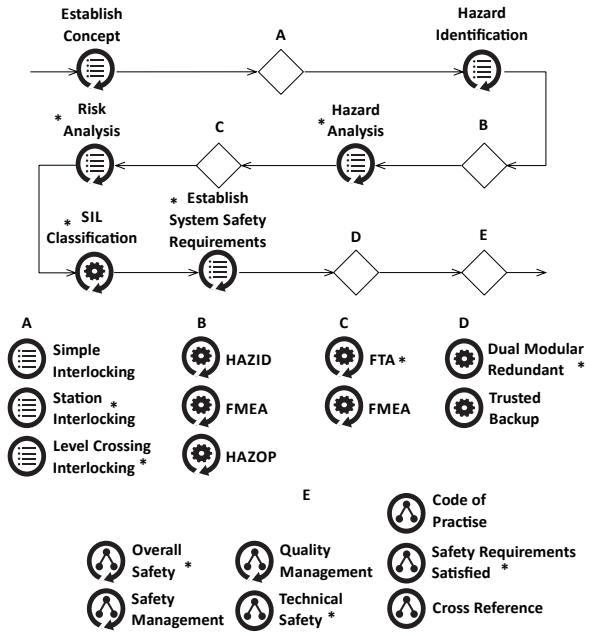

Figure 3. Pattern Selection

V. Elicit Functional Requirements

A. Pattern Selection

Fig. 3 provides an overview of the patterns considered in the railway interlocking case, organised into a pattern selection map based on defined relationships between patterns. Due to space restrictions the pattern definitions are not provided (described in [12]). The following abbrevations are used in Fig. 3: HAZID – HAZard IDentification; FMEA – Failure Modes and Effects Analysis; HAZOP – HAZard and OPerability Studies; FTA – Fault Tree Analysis; SIL – Safety Integrity Level.

The selection process starts at the pattern referenced in the upper left corner of Fig. 3 and follows the direction of the arrows. Pattern selection ends when all patterns have been considered. The diamond represents a choice; more than one pattern may be selected. The letter above a choice is a reference to a correspondingly named group of patterns in the lower part of Fig. 3. The symbol "*" is used to identify the patterns that are used in this article. Each pattern definition clearly describes the problem addressed by the pattern and its intended application, thus the pattern definitions may be conferred for support in the selection process. The application of the SaCS method and the selection process as exemplified in this article assures that relevant patterns may be selected at each stage of development. The rationale given by a user at each selection step on the selection of patterns should give assurance for the correct set of patterns being selected.

We assume in this article that the information provided in Section II sufficiently details the development objectives such that the pattern *Establish Concept* (supports clarifying objectives) may be passed in the selection process.

The patterns *Station Interlocking* and *Level Crossing Interlocking* associated with choice A in Fig. 3 capture the problem of eliciting functional requirements for a system that shall control the appliances available in a station with several tracks and a level crossing, respectively. They were chosen as support.

B. Pattern Instantiation

A pattern may have multiple input and output parameters. The instantiation of the input parameters define the context for interpreting the pattern, while the instantiation of the output parameters define the result of pattern instantiation. The pattern definition describes the transition from input to output. The instantiation of the pattern *Station Interlocking* produced a set of requirements. The following is one of these requirements.

"FR.1: a train route AX may be locked (secured for train movements) when: a) the train routes AY, M, O, N, BX and BY are in the state not locked; and b) point P1 is aligned; and c) track sections A, X and B are in the state vacant."

C. Pattern Composition

Fig. 4 specifies a composite pattern. Everything above the horizontal line may be thought of as a kind of preamble. The icon for a composite occurs in the upper left corner underneath the name of the composite, which is *Functional Requirements*. The inputs and outputs of the composite are tagged by an arrow pointing either towards (indicating input) or from (indicating output) a list of parameters. The parameter list is visualised on the form [<parameter list>]. Everything below the horizontal line describes the actual composition. The input and output of the patterns occurring within the composite are distinguished in the same manner as input and output in the preamble. The input parameter Mch of the composite is used by both contained patterns.

The patterns *Station Interlocking* and *Level Crossing Interlocking* are related by the *combines* operator (symbolised by two overlapping circles). Hence, the icon decorating the line connecting the parameter lists in Fig. 4 symbolises a combines relationship.

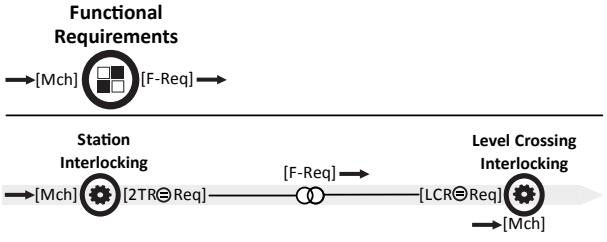

Figure 4. Functional Requirements – Composite

The annotation [2TR=Req] (the symbol = enclosed by a circle represents an *alias* operator) found in Fig. 4 defines an alias 2TR for the output parameter Req. The *combines* operator creates an output parameter list named F-Req that consists of 2TR and LCR.

The grey wide arrow in the background indicates the recommended pattern instantiation order and gives guidance to the process of applying the patterns.

VI. Elicit Safety Requirements

A. Pattern Selection

Once the main functional requirements have been elicited based on selected patterns from choice A of Fig. 3, further traversal leads to the pattern *Hazard Identification*. This pattern defines the process of identifying potential hazards. In choice B, patterns describing methods for hazard identification are offered. We assume that the hazards associated with the operation of the interlocking system, e.g., collision train-train, collision train-object, and level crossing accident, are identified such that the *Hazard Identification* pattern as well as the patterns in choice B may be passed in the selection process.

The *Hazard Analysis* pattern however is selected as it provides guidance on the process of deriving the potential causes of hazards. In choice C of Fig. 3, different process solution patterns (representing methods) supporting hazard analysis may be selected. The *FTA* was selected as support for *Hazard Analysis* under the assumption that a top-down fault tree analysis is an acceptable and effective method for identifying potential causes of failure.

Further traversal of Fig. 3 leads to the pattern *Risk Analysis*. The pattern provides guidance on how to address identified hazards and to establish a notion of risk. The *SIL Classification* pattern was selected as it defines the method for classifying railway systems and their components with respect to criticality.

The pattern *Establish System Safety Requirements* describes the process of establishing safety requirements based on inputs from risk assessment. It was regarded as relevant for the case and selected as support.

B. Pattern Instantiation

Safety requirements are defined on the basis of risk assessment. The process requirement patterns selected in Section VI-A support the process of eliciting safety requirements and may be applied subsequently in the following order:

1. *Hazard Analysis* – used to identify potential causes of hazards based on input on applicable hazards.
2. *Risk Analysis* – used for addressing hazards with respect to their severity and likelihood of occurring combined into a notion of risk.
3. *Establish System Safety Requirements* – used for defining requirements on the basis of identified risks.

When *Establish System Safety Requirements* was instantiated on the basis of inputs provided by the instantiation of its successors, the following safety requirement was among those identified: *"SR.1: A main signal belonging to a train route may only signal a proceed aspect if the train route is locked"*.

Other results from the instantiation of the mentioned patterns were a hazard log that traces identified hazards to potential causes of these hazards, a fault tree analysis, and a qualitative risk assessment. In the following we only refer to the safety requirement exemplified above.

C. Pattern Composition

Fig. 5 presents the *Safety Requirements* composite. The pattern has two input parameters namely ToA (short for Target of Assessment) and Haz (short for Hazards) and one output parameter S-Req (short for Safety Requirements).

The hazards associated with the parameter Haz, e.g., collision train-train and collision train-object, consisted of a set of relevant generic top events. The hazards were assessed with respect to the intended operation of interlocking system by the use of the *Hazard Analysis* pattern, supported by the *FTA* pattern. The result of pattern instantiation is the output HzLg (short for hazard log) containing an overview over hazards and their potential causes.

The output HzLg from *Hazard Analysis* is an input of the *Risk Analysis* pattern (illustrated by the *assigns* operator that is drawn as an arrow from HzLg to Haz). The *SIL Classification* pattern was used as support for *Risk Analysis*. The output Risks from *Risk Analysis* is used as input to *Establish System Safety Requirements*.

The output S-Req of *Establish System Safety Requirements* (S-Req is defined as an alias for the output parameter Req) is an output of the composite.

Fig. 6 defines the *Requirements* composite. The composite is defined in order to make later illustrations simpler and consists of the composite for establishing functional requirements (defined in Fig. 4) and the composite for establishing

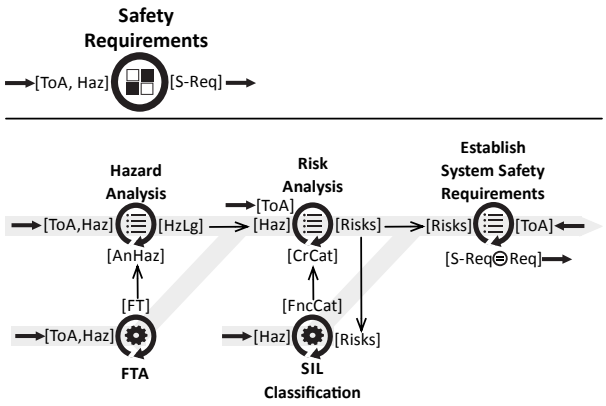

Figure 5. Safety Requirements – Composite

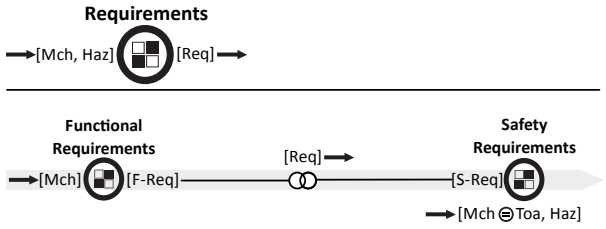

Figure 6. Requirements – Composite

safety requirements (defined in Fig. 5).

VII. Establish Design Basis

A. Pattern Selection

At choice D of Fig. 3, there are two available design patterns that may be selected.

The *Trusted Backup* pattern describes a system concept where an adaptable controller may operate freely in a delimited operational state space. Safety is assured by a redundant non-adaptable controller that operates in a broader state space and in parallel with the adaptable controller. A control delegator grants control privileges to the most suitable controller at any given time on the basis of switching rules and information from safety monitoring.

The *Dual Modular Redundant* pattern describes a system concept where two similar controllers operate in parallel. The parallel operating redundant controllers and a voting unit provides mitigations against random error.

The *Dual Modular Redundant* pattern was selected as guidance for establishing the design on the basis of an evaluation of the strengths and weaknesses of the two design patterns with respect to the requirements.

B. Pattern Instantiation

Fig. 7 represents an excerpt (simplified) of the result, fully described in [12], of applying the *Dual Modular Redundant* pattern. The pattern was instantiated according to its instantiation rule and the design was defined to comply with the requirements identified in Section V and Section VI.

Fig. 7 illustrates the main parts of our interlocking system. A component identified as *Cmd* is responsible for the communication towards the operators of the system. Dual controllers, identified as *Ctrl1* and *Ctrl2*, are responsible for providing interlocking functionality, e.g., lock a train route upon request from an operator. A component identified as *IO* is responsible for communicating with e.g., points, lights, and track sections as presented in Fig. 1 in order to communicate their states to the dual controllers. The *IO* component is also responsible for safe application of the output from the dual controllers (the *IO* component contains a voter).

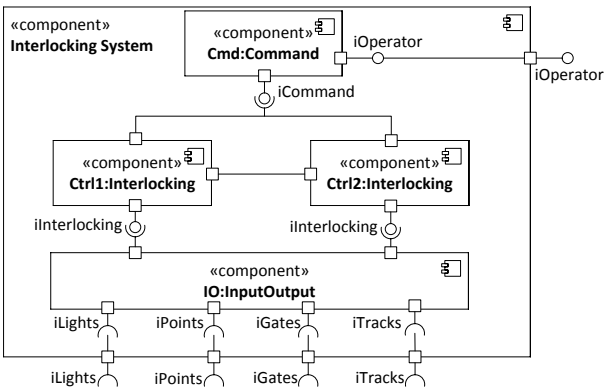

Figure 7. UML Component Diagram

Besides the presented excerpt, results from pattern instantiation include a description of: the interaction between the interlocking system and other systems; the functionality of the internal components of the interlocking system; and the interaction between the components within the interlocking system. In the following when we refer to the system design we mean the full design of the interlocking system. The fulfilment of identified requirements, e.g., FR.1 defined in Section V-B and SR.1 defined in Section VI-B, is manifested in different decomposed models of the full design.

C. Pattern Composition

In Fig. 8, the output parameter S of *Dual Modular Redundant* represents the abstract system design described by the pattern, the instantiation of which is represented by the design outlined in Section VII-B.

Fig. 8 specifies that the instantiation of the output parameter S of the pattern *Dual Modular Redundant* shall satisfy the instantiation of the output parameter Req of the pattern *Requirements*. The relationship is captured by a *satisfies* operator where the bullet is associated with the output that describes what should be satisfied and the check mark is associated with the output that is required to satisfy.

VIII. Establish Safety Case

A. Pattern Selection

The *Overall Safety* pattern associated with choice E in Fig. 3 was selected as a means to argue that the system is

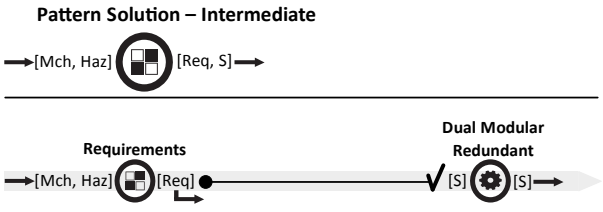

Figure 8. Intermediate Solution – Composite

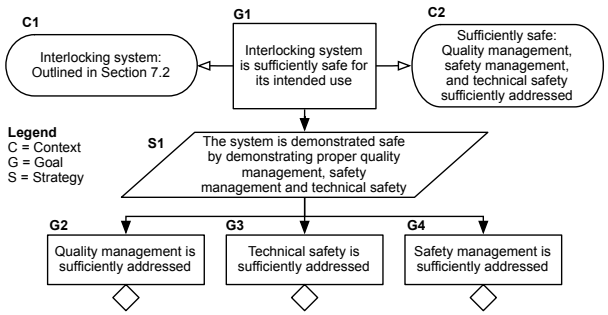

Figure 9. GSN Safety Case (excerpt)

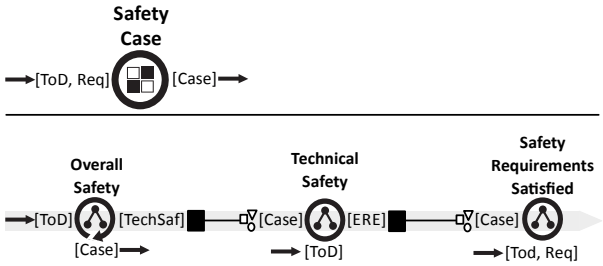

Figure 10. Safety Case – Composite

sufficiently safe based on satisfactory quality management, safety management and technical safety.

The *Technical Safety* pattern was selected for arguing satisfactory technical safety. A strategy defined by the pattern is to explicitly address all risks associated with the system (contrary to an implicit safety demonstration). The explicit risk demonstration strategy is here intended to be detailed by the *Safety Requirements Satisfied* pattern that may be used to structure the argument that all requirements are satisfied.

B. Pattern Instantiation

Fig. 9 presents an excerpt expressed in GSN [13] of the safety case provided upon instantiation of *Overall Safety*, *Technical Safety* and *Safety Requirements Satisfied* according to their instantiation rules.

The safety case provided upon pattern instantiation describes a decomposable safety demonstration, annotated in GSN [13], arguing that the system design is sufficiently safe for its intended purpose. One of the decomposed parts of the safety case put forward a claim that the system is safe given that the safety requirements are correct, suitable and satisfied. Further, compliance to identified safety requirements is shown by referring to the properties of the design as defined by the design models. The design models act as supporting evidences to claims put forward in the safety case. The safety case contains the claims that must be supported by evidences and that once shown provides assurance that the design is safe.

C. Pattern Composition

Fig. 10 defines the composite *Safety Case*. It specifies that the input to the composite is ToD (short for Target of Demonstration) and Req (short for Requirements). The output of the composite is identified as Case and represents the output provided when the pattern *Overall Safety* is instantiated.

A part identified as TechSaf (short for Technical Safety) of the *Overall Safety* patterns is detailed (specified by the *details* operator, the "black box" represents the output that is detailed and the small icons represent the output that details) by the *Technical Safety Pattern*. A part identified as ERE (short for Explicit Risk Estimation) of *Technical Safety* is detailed by *Safety Requirements Satisfied*.

IX. Composite Pattern Solution

A. Pattern Composition

Fig. 11 defines the composite *Pattern Solution* that combines all patterns applied in the case.

Using the *satisfies* operator *Pattern Solution* specifies that the instantiation of output parameter S of the *Dual Modular Redundant* shall satisfy the instantiation of Req of the composite *Requirements* (defined in Fig. 6). Further, the *demonstrates* operator constrains the instantiation of the output parameter Case of *Safety Case* to be a safety demonstration for S of *Dual Modular Redundant*. It is also specified that the instantiation of parameter S-Req (representing an element of the parameter set named Req, see Fig. 6) of the composite *Requirements* is assigned to the input parameter Req of *Safety Case*. The instantiation order of the patterns is defined by the grey arrow in the background that indicates that the *Requirement* composite shall be instantiated first, then the *Dual Modular Redundant* and the *Safety Case* patterns may be instantiated in parallel.

B. Pattern Instantiation

The composite *Pattern Solution* described in Fig. 11 is a pattern and may thus be applied on several cases, e.g., for developing an interlocking system for a station similar

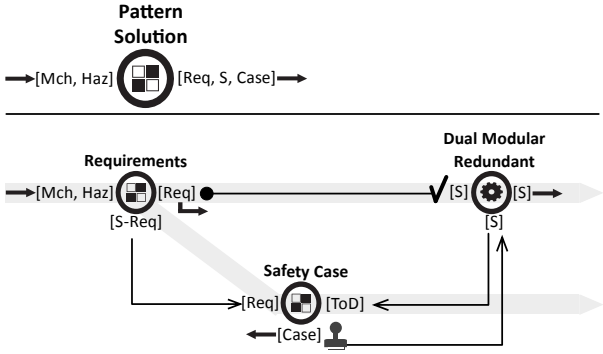

Figure 11. Pattern Solution – Composite

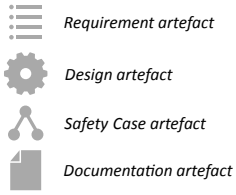

Figure 12. Icons for denoting different types of artefacts

to the one described in this article. In order to describe a specific application of a composite pattern, annotations for specifying the instantiation of parameters may be added to the composite pattern diagram.

Fig. 12 illustrates the different icons and the respective type of artefacts that they represent that are used to specify parameter instantiation.

Fig. 13 is identical to Fig. 11 with the addition of annotations specifying the instantiations of parameters. An icon symbolising the type of artefact that is referred to and a string identifying the referred artefact illustrates an artefact reference. A dotted line connecting an artefact reference to a parameter specifies that the referred artefact instantiates the parameter.

The instantiations illustrated in Fig. 13 refer to the artefacts that are described in this article rather than the full version of these artefacts as provided in [12].

When every composite pattern diagram that is applied during development is annotated with their instantiations, the traceability between the input and output of every pattern and between patterns are provided.

X. Related Work

To the best of our knowledge, there exists no other pattern-based method that combines diverse kinds of patterns into compositions like SaCS. SaCS has been conceived to facilitate efficient development by the support of patterns, separation of concerns in the style of pattern languages, a clearly defined process and application of acceptable development practices as required by safety standards, and at the same time documentation and visualisation in the manner of system modelling.

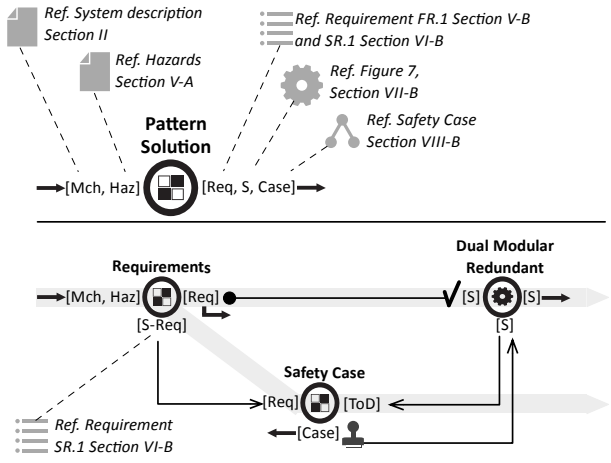

Figure 13. Pattern Solution – Composite (annotated with instantiations)

The concept of systematically applying a set of patterns is inspired by the work of Alexander et. al [6] on architecture of buildings. Important sources of inspiration applicable to development of software based systems are pattern approaches for: requirements elicitation [14], [15], software design [7], [8], [16], and safety demonstration [17], [13]. Two challenges associated with the referenced pattern approaches are that: the integration of patterns is detailed informally; each pattern approach only reflects on one perspective important when developing critical systems, e.g., software design [8]. SaCS offer the ability to combine different kinds of patterns and detail the combination.

The notation for detailing the application of patterns and the focus on establishing a complementing set of development artefacts offered by SaCS are inspired by safety domain needs on providing assurance. International safety standards, e.g., [2], [4] express requirements related to assurance. While safety standards to a large degree describe what are the required practices to be performed during development, SaCS provides guidance on applying accepted safety engineering practices. The European railway regulation [18] and the associated guideline [19] on common safety methods for risk evaluation and assessment has influenced the work on defining SaCS patterns. While the guideline [19] addresses the railway domain, SaCS may be applied in different domains by selecting among the available patterns those that are accepted within a given domain.

XI. Conclusions

We have demonstrated how the conceptual design is instantiated from several SaCS basic patterns within a specific SaCS composite (Fig. 13). Each constituent basic pattern of the composite has clearly defined inputs and outputs and provides guidance on instantiation through defined instantiation rules (these are detailed in [12]). Operators for composition define the combination of instantiation results from several patterns. The composite pattern details: the identifier and type of every pattern applied in order to derive the conceptual design; the pattern instantiation order; and the flow of data through the network of patterns giving traceability between development artefacts.

The conceptual design is built systematically in manageable steps (exemplified in Sections V to IX) by a clearly defined process for pattern selection, instantiation and merging of results (by pattern composition). The conceptual design (fully described in [12]) is consistent (see Section IV). The required triple is provided by the instantiation of the parameters *Req, S*, and *Case* of the composite pattern

 ISBN: 978-1-61208-268-4

illustrated in Fig. 13. In [12] we argue that the triple completely specifies the required function. We also argue that the design (instantiation of S in Fig. 13) correctly specifies the fulfilment of requirements (instantiation of Req in Fig. 13). The conceptual design is expressed in a form that we think is easy to understand (textual descriptions, UML diagrams, and GSN [13] diagrams) and that is easy to detail or reuse. We also claim that the conceptual design may be easily detailed into an implementable system.

XII. Acknowledgments

This work has been conducted and funded within the OECD Halden Reactor Project, Institute for energy technology (IFE), Halden, Norway.

References

[1] A. A. Hauge and K. Stølen, "A pattern-based method for safe control systems exemplified within nuclear power production," in *Lecture Notes in Computer Science*, vol. 7612, September 2012, pp. 13–24.

[2] IEC 62278, "Railway Applications – Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)," IEC 62278, ed. 1.0, International Electrotechnical Commission, 2002.

[3] IEC 62279, "Railway Applications – Communication, Signalling and Processing Systems – Software for Railway Control and Protection Systems," IEC 62279, ed. 1.0, International Electrotechnical Commission, 2002.

[4] IEC 62425, "Railway Applications – Communication, Signalling and Processing Systems – Safety Related Electronic Systems for Signalling," IEC 62425, ed. 1.0, International Electrotechnical Commission, 2007.

[5] IEC 61025, "Fault Tree Analysis (FTA)," IEC 61025, ed. 2.0, International Electrotechnical Commission, 2006.

[6] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977.

[7] F. Buschmann, K. Henney, and D. Schmidt, *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages, Vol. 5*. Wiley, 2007.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[9] OMG, "Unified Modeling Language Specification, version 2.4.1," Object Management Group, 2011.

[10] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis: The CORAS Approach*, 1st ed. Springer, 2010.

[11] E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Graphics Press, 5 2001.

[12] A. A. Hauge and K. Stølen, "A Pattern Based Method for Safe Control Conceptualisation – Exemplified Within Railway," Institute for energy technology, OECD Halden Reactor Project, Halden, Norway, Tech. Rep. HWR-1037, 2013.

[13] GSN Working Group, "GSN Community Standard Version 1," York, England, 2011.

[14] E. Hull, K. Jackson, and J. Dick, *Requirements Engineering*. Springer, 2004.

[15] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley, 2001.

[16] R. S. Hanmer, *Patterns for Fault Tolerant Software*. Wiley, 2007.

[17] R. Alexander, T. Kelly, Z. Kurd, and J. McDermid, "Safety Cases for Advanced Control Software: Safety Case Patterns," University of York, York, UK, 2007.

[18] European Union, "Commission regulation (EC) No 352/2009 on the adoption of a common safety method on risk evaluation and assessment as referred to in Article 6(3)(a) of Directive 2004/49/EC of the European Parliament and of the Council," Official journal of the European Union, 2009.

[19] ERA, "Guide for the application of the Commission Regulation on the adoption of a common safety method on risk evaluation and assessment as referred to in Article 6(3)(a) of the Railway Safety Directive," European Railway Agency, 2009.

 ISBN: 978-1-61208-268-4