SACS — A Pattern Language for Safe Adaptive Control Software

André Alexandersen Hauge,
Department of ICT Risk and Dependability, Institute for energy technology, Halden, Norway
Department of Informatics, University of Oslo, Norway

Ketil Stølen,
Department of Networked Systems and Services, SINTEF ICT, Oslo, Norway
Department of Informatics, University of Oslo, Norway

This article puts forward a pattern language for Safe Adaptive Control Software named SACS. We interpret the term "pattern language" such that a set of patterns, the interconnections between them, and how these are intended to be used make up a language. The pattern language consists of three basic types of patterns, namely the *Requirement Pattern*, the *Design Pattern*, and the *Safety Case Pattern* and an additional *Composite Pattern* type. The Composite Pattern type facilitates users of the language to specify their own patterns as compositions of basic patterns. The patterns are intended to be used in the context of safety related and safety critical systems, thus the safety aspect is a principal concern. The pattern language may be used as a tool for e.g. safety engineers and system developers to increase effectiveness during conceptual design and facilitate effective evaluation of alternative adaptive design solutions with respect to utilisation in a safety related application by a systematic approach for combining best practices.

Categories and Subject Descriptors: D.2.10 [**Software Engineering**]: Design—*Representation*

General Terms: Design, Documentation

Additional Key Words and Phrases: pattern language, adaptive control software, safety assurance

ACM Reference Format:

Hauge, A.A. and Stølen, K. 2011. SACS — A Pattern Language for Safe Adaptive Control Software. 18th Conference on Pattern Languages of Programs (PLoP), Portland, Oregon, USA (October 2011), 22 pages.

1. INTRODUCTION

Adding adaptive behaviour to the features of a control system increases the complexity as well as the potential risk of erroneous behaviour. As it is reasonable to assume that the objective for adding adaptive behaviour is to yield a positive net effect and at the same time uphold the safety integrity compared to a traditional control system, we may derive the following general requirements:

i) the adaptive control feature shall provide a level of performance (or increased level of safety with respect to events that require a resilient handling) which may not be feasible with conventional control techniques;

ii) the adaptive control feature shall have no negative effect on safety (or decrease level of safety with respect to events that are normally mitigated by conventional control techniques).

This work has been conducted and funded within the OECD Halden Reactor Project, a nuclear research project managed by Institute for energy technology (IFE), Halden, Norway.
1. Author's address: André A. Hauge, Institute for Energy Technology, Os Alle 5, NO-1777, Halden, Norway; email: andre.hauge@hrp.no;
2. Author's address: Ketil Stølen, SINTEF ICT, P.O.Box 124 Blindern, N-0314 Oslo, Norway; email: ketil.stolen@sintef.no;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission. A preliminary version of this paper was presented in a writers' workshop at the 18th Conference on Pattern Languages of Programs (PLoP). PLoP'11, October 21-23, Portland, Oregon, USA. Copyright 2011 is held by the author(s). ACM 978-1-4503-1283-7

The requirements (i) and (ii) may be satisfied by different means and strategies. Requirement (i) put constraints on the provision of a justification for introducing adaptivity and on providing the confidence that a higher level of performance may be obtained and by what means. Requirement (ii) put constraints on the need to provide a safety demonstration. In this article we will focus on the safety aspect. Three possible approaches for satisfying (ii) may be:

I) to constrain the adaptation mechanism such that the adaptable controller may not evolve into an unsafe configuration;

II) to evaluate at runtime that any adaptation is safe before committing a modification of the control software;

III) to constrain the set of operational states for which the adaptive controller is granted control privileges rather than constrain the adaptation mechanism.

In this article, we address the challenges associated with the approach outlined in alternative (III). We specify three complementary types of patterns and how they may be combined in order to augment a pattern language solution. The patterns represent an initial set of patterns intended as parts of an extensible pattern language.

An overview of the SACS language is given in Section 2. Section 3 presents a set of SACS patterns. Section 4 describes the use of the presented patterns in the context of an example railway interlock system. Section 5 concludes and describes future work.

2. THE SACS PATTERN LANGUAGE

The UML class diagram [Object Management Group (OMG) 2010] in Figure 1 illustrates the main elements available in SACS.

There are three ***Basic Pattern*** types available in SACS, each with its own specialisation added to their common format. The ***Requirement Pattern*** type is inspired by the problem frames approach [Jackson 2001]. The problem frames approach is a means to specify the problem domains and interaction phenomena between problem domains in order to derive requirements for the system which is intended to be built. The ***Design Pattern*** type is inspired by the well-known GoF[1] [Gamma et al. 1995] approach to describe design solutions, whereas the ***Safety Case Pattern*** type focuses on providing solutions for how to demonstrate that a system is sufficiently safe for its intended purpose. The goal of the *Safety Case Pattern* is to provide an approach to the structure of claims such that it may be logically deduced and concluded that the system, built according to a SACS pattern, is sufficiently safe.

The ***Composite Pattern*** type is provided as a means for a user to detail how a set of best practices (here predefined basic SACS patterns) may be combined to solve a specific design problem. The notion of a user defined pattern is somewhat different from the traditional view on what a pattern shall be. Besides that a Composite Pattern is given a name, the traditional parts of a pattern format is exchanged with a notation to reference and combine basic patterns that are detailed in a more traditional manner.

The notion of a ***Pattern Artefact*** is an important ingredient in SACS. Figure 1 assigns an artefact type to each basic pattern type. The pattern artefact type ***Requirement*** encapsulates the set of requirements contained in a Requirement Pattern. The ***Specification*** type represents the abstract design specification contained within a Design Pattern. The ***Demonstration*** type represents the argument structure contained within a Safety Case Pattern.

In order to detail in a *Composite Pattern* how the contained parts of a target pattern may be used as parameters in a source pattern, a set of combinators are defined. A part is here meant a pattern artefact. Each ***Combinator*** is directed

[1]The GoF abbreviation is short for Gang of Four and refers to the four authors of the book [Gamma et al. 1995].

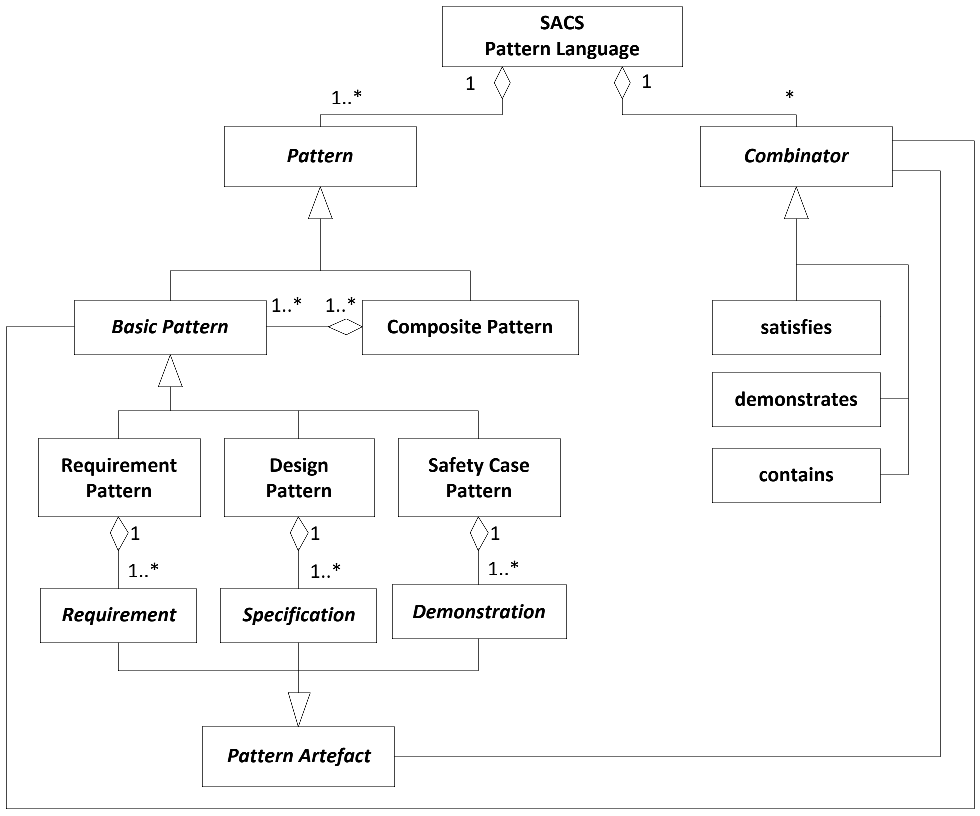

Fig. 1. The Basic Elements of SACS

and relates a pattern (its source) to a pattern artefact (its target). The semantics of the combinators may be summarised into:

contains — express that a pattern embodies a specific pattern artefact. When specifying a Composite Pattern, the *contains* combinator is used to explicitly state that a distinct pattern artefact (of the type Requirement, Specification or Demonstration) is contained in a source pattern.

satisfies — express that a Design Pattern shall satisfy a Requirement Pattern Artefact. More concretely, the Design Pattern is bound to a pattern artefact entity of type *Requirement* which is defined as contained in a Requirement Pattern by the *contains* relation. The *satisfies* combinator identifies which requirements that shall be satisfied.

demonstrates — express that a Safety Case Pattern may be used to provide a systematic safety demonstration of a system developed according to the principles outlined in a Design Pattern. More concretely, the Safety Case Pattern is bound to a pattern artefact entity of type *Specification* which is defined as contained in a Design Pattern by the *contains* relation.

A graphical notation is provided to support the specification of Composite Patterns. Figure 2 illustrates the main graphical elements of SACS. A short textual description is given below.

a) ***Composite Pattern Diagram***: a solid black diagram frame with a frame label, a parameter box, and a compartment. The frame label consist of the keyword **cmp** and a user given name. The parameter box is used to hold the required pattern artefacts to be given when instantiating the pattern. The general format for detailing parameters in a parameter

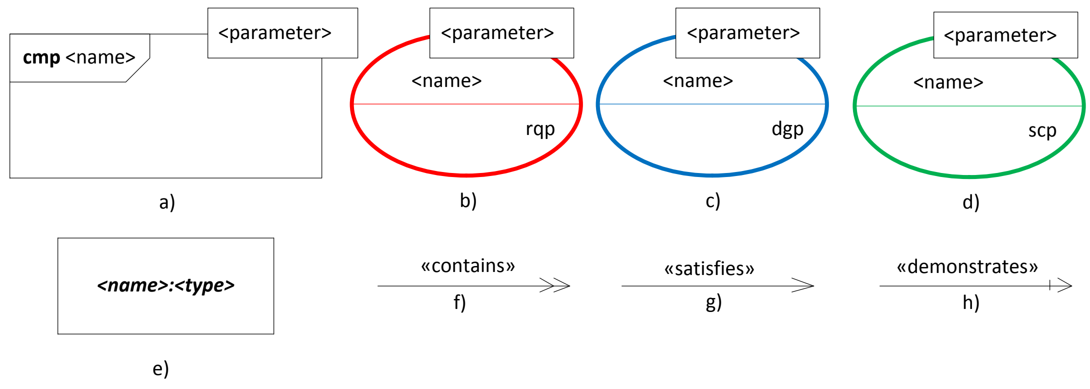

Fig. 2. Graphical Notation of SACS

box is <short name>":"<type>. In the event of more than one parameter then *short name* is represented by a comma separated list of identifiers. The *type* field is used to denote the pattern artefact type. The compartment is used to hold the user defined composition of patterns;

b) ***Requirement Pattern Reference***: a graphical reference to a Requirement Pattern where the name of the referenced pattern is provided in the upper department of a solid line red oval. The bottom compartment of the oval is marked with the keyword **rqp**;

c) ***Design Pattern Reference***: a graphical reference to a Design Pattern where the name of the referenced pattern is provided in the upper department of a solid line blue oval. The bottom compartment of the oval is marked with the keyword **dgp**;

d) ***Safety Case Pattern Reference***: a graphical reference to a Safety Case Pattern where the name of the referenced pattern is provided in the upper department of a solid line green oval. The bottom compartment of the oval is marked with the keyword **scp**;

e) ***Pattern Artefact***: illustrated in a similar manner as an abstract class in UML. The class may be given a name and its type may be specified. Contained elements within a pattern artefact may be specified as properties are specified in a UML class.

f) ***Contains Relation***: an arrow annotated with the keyword **«contains»**. Used to denote that a *Pattern Artefact* is contained in a Pattern.

g) ***Satisfies Relation***: an arrow annotated with the keyword **«satisfies»**. Used to combine a *Design Pattern* and a *Requirement* artefact associated with a *Requirement Pattern*.

h) ***Demonstrates Relation***: an arrow annotated with the keyword **«demonstrates»**. Used to combine a *Safety Case Pattern* and a *Specification* artefact associated with a *Design Pattern*.

3. SACS PATTERNS

A pattern in SACS may be a basic pattern or any composition of patterns framed within a Composite Pattern description. Thus a basic pattern may be used standalone or it may be used in combination with other patterns. Four patterns will be presented in this article, one *Composite Pattern* and three *Basic Patterns*. The first of these patterns, presented in Figure 3, is a composite pattern named *Extended Trusted Backup*. The composite pattern is composed of three basic patterns, one for each of the three available basic pattern types *Requirement Pattern*, *Design Pattern* and *Safety Case Pattern*. Section 3.1 provides an overview of the *Extended Trusted Backup* composite pattern and shows by example how a set of basic patterns may be bound to each other. The basic patterns that are referenced and used in the composition are defined individually in the Sections 3.2, 3.3 and 3.4.

3.1 A Composite Pattern Example

We specify the relationships between a set of basic patterns as a *Composite Pattern* in Figure 3 by the notation introduced in Section 2. The composite pattern (Figure 3) is complete in the sense that it contains a basic pattern of each of the three types. The basic integration idea may be summarised as follows:

—The *Extended Trusted Backup* composite pattern takes *Operator* and *Plant* as parameters. The specification of parameters at the Composite Pattern level is the result of extracting the set of required pattern artefacts (the parameters) specified for its constituent Requirement Pattern. The basis for defining parameters of the *Extended Trusted Backup* then is the parameters of the requirement pattern named *State Space Subset*.

—The Requirement Pattern is parametrised in order to elaborate upon the problem of adaptive control with respect to a specific context. The specific context of interest with respect to the *State Space Subset* requirement pattern is given by the parameters *Operator* and *Plant*. These parameters represent the parts of the world that the system under development will interact with and is given or detailed once a user decides upon the context for which the pattern is applied. The *Operator* and *Plant* parameters represent the human operator interacting with the adaptive system and the process or equipment to be controlled by means of adaptive control respectively.

—The *Requirement* artefact, defined as a part of the *State Space Subset* requirement pattern, contains a set of requirements for each named designed problem domain part of the pattern. Each requirement set is named and contains requirements addressing the required function or constraints associated with the respective named part. These named parts represent parts that are assumed to be represented in any system solution that intends to satisfy the requirement pattern. Examples of named parts specified in the *State Space Subset* pattern are the *ac (Adaptive Controller)*, the *tc (Trusted Controller)* and the *cd (Control Delegator)*. The naming format is <short name> "("<long name>"):"<type>. As a result of instantiating the requirement pattern, the named parts may be given more descriptive names and the generic requirements may be concretised and further detailed. In order to provide detailed linking, a common naming scheme is used where that is appropriate. As an example, the *Adaptable Controller* entity appearing in the *State Space subset* requirement pattern (Section 3.2) is intended to be linked with the *Adaptable Controller* entity as described in the *Trusted Backup* design pattern (Section 3.3). These entities, appearing in different patterns,

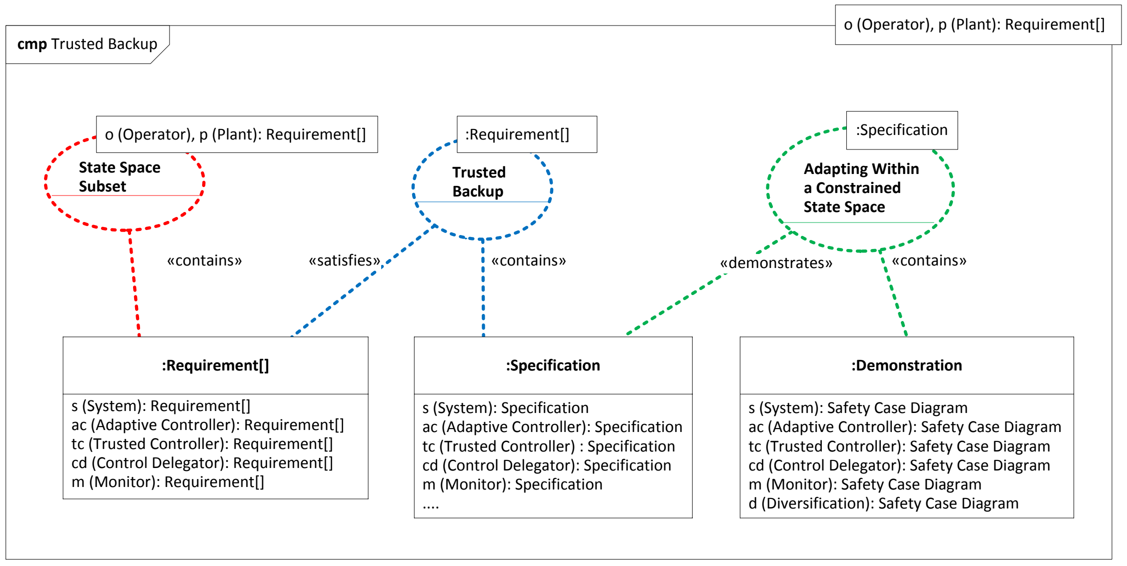
cmp Trusted Backup
o (Operator), p (Plant): Requirement[]
o (Operator), p (Plant): Requirement[]
State Space Subset
:Requirement[]
Trusted Backup
:Specification
Adapting Within a Constrained State Space
«contains»
«satisfies»
«contains»
«demonstrates»
«contains»
:Requirement[]
s (System): Requirement[]
ac (Adaptive Controller): Requirement[]
tc (Trusted Controller): Requirement[]
cd (Control Delegator): Requirement[]
m (Monitor): Requirement[]
:Specification
s (System): Specification
ac (Adaptive Controller): Specification
tc (Trusted Controller) : Specification
cd (Control Delegator): Specification
m (Monitor): Specification
....
:Demonstration
s (System): Safety Case Diagram
ac (Adaptive Controller): Safety Case Diagram
tc (Trusted Controller): Safety Case Diagram
cd (Control Delegator): Safety Case Diagram
m (Monitor): Safety Case Diagram
d (Diversification): Safety Case Diagram

Fig. 3. Pattern Integration and Binding

represent the same entity but are addressed from different perspectives. The *Adaptable Controller* entity represents a problem domain when appearing in the requirement pattern whereas it represents a system part when appearing in the design pattern. The *Adaptable Controller* also appears in the safety case pattern (Section 3.4), but then as a contextual element.

—The Design Pattern named *Trusted Backup* contains a specification of the structure or behaviour of parts that corresponds in name to the named parts of the *Requirement* artefact. Thus a named part that is detailed in a Design Pattern should satisfy the requirements of the correspondingly named part of the *Requirement* artefact (or otherwise linked). In Figure 3 this is shown with a *satisfies* relationship to a *Requirement* artefact. The specification of parameters in the parameter box of the *Trusted Backup* design pattern may them be reduced to only specify the type of the required pattern artefact. The concrete parameters are detailed by the bound class in the listing provided in its lower compartment.

—The Design Pattern provides an artefact of type *Specification*. It is assumed that the *Specification* obtained from instantiating the Design Pattern will contain important information regarding e.g. operating context, what is acceptably safe system behaviour within its intended context, besides pure design issues.

—The Safety Case Pattern uses the *Specification* artefact as an input to put the safety *Demonstration* described in the pattern in its proper context.

3.2 State Space Subset: Requirement Pattern

Name: State Space Subset.

Intent: The intent is to address the basic problem domains associated with a control system which employ a strategy where an adaptable controller is only allowed to operate in a limited part of the operational state space. The strategy involves dividing the state space of what is to be controlled into a set of overlapping operational regions (partitions) and use redundant and alternative means of control in order to reduce risk associated with adaptive controller failure.

Motivation: The motivation for applying the pattern is when the adaptive control feature may provide an optimisation effect in a limited part of the operational state space and where there exist alternative means of control which may be applied in order to protect against or mitigate erroneous adaptive behaviour. Given such a problem, the state space may be divided into control regions where safety must be guaranteed and regions where an unconstrained adaptive controller will not negatively affect safety and thus may be allowed to operate.
An example of such a problem may be an adaptive flight control system such that the flight performance of an aeroplane might be improved or provided with resilience to unanticipated events. A representative case is presented in [Schumann and Liu 2010]. A set of limit values on the pitch, roll and yaw rotation along their respective axis with respect to the plane's centre of mass may be used to delimit the state space for which adaptive control operations may be applied. A region of the state space could then be specified by defining a set of rules which detail the values for the pitch, roll and yaw rotation respectively for which adaptive control may be allowed. In the remaining part of the overall operating region, adaptive control would not be allowed such that a standard flight control system should also be available.

Applicability: The pattern is suitable to apply in the following situations:

—when it is assumed that an operational state space may be divided into diverse operational regions;

—when it is assumed that adaptive control may provide optimisation of operations within a region of an operational state space;

—when it is assumed that a safe fall-back system may mitigate unwanted operations occurring as an effect of erroneous adaptive controller.

Related Patterns: The *Trusted Backup* Design Pattern has a *satisfies* relationship to the *State Space Subset* Requirement Pattern which details the main requirements which must be satisfied by design.

Safety Classification: General requirements.

Problem Context: The Problem Diagram (denoted by the **PD** abbrevation in the upper left corner of the frame) provided in Figure 4 illustrates the main problem domains which play a part in the pattern.

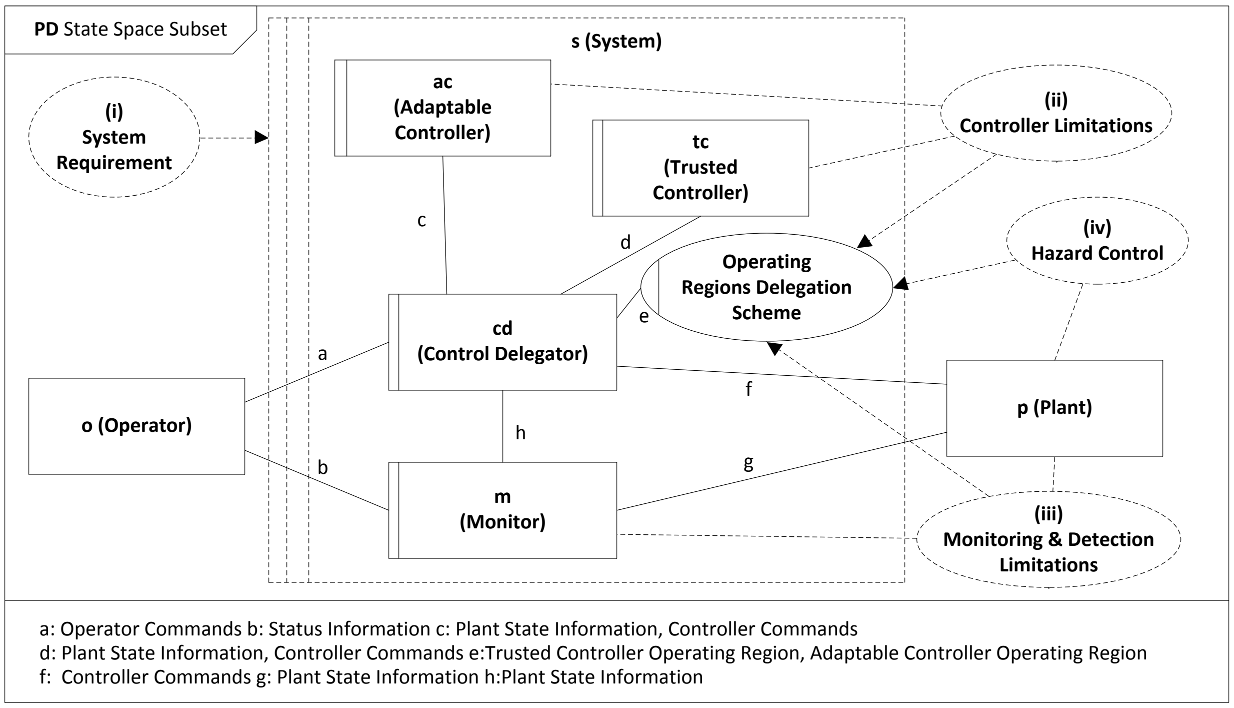
PD State Space Subset
s (System)
(i) System Requirement
ac (Adaptable Controller)
tc (Trusted Controller)
(ii) Controller Limitations
Operating Regions Delegation Scheme
(iv) Hazard Control
cd (Control Delegator)
o (Operator)
p (Plant)
m (Monitor)
(iii) Monitoring & Detection Limitations
a
b
c
d
e
f
g
h
a: Operator Commands b: Status Information c: Plant State Information, Controller Commands
d: Plant State Information, Controller Commands e:Trusted Controller Operating Region, Adaptable Controller Operating Region
f: Controller Commands g: Plant State Information h:Plant State Information

Fig. 4. Problem Diagram - State Space Subset

The different diagram entities are defined according to the problem frames notation [Jackson 2001] and may be interpreted as follows:

—*Operator* represents a given biddable problem domain in the form of a human operator which interacts with the system;
—*Adaptable Controller* represents a designed part of the system (designed problem domain) which provides control functionality and which incorporates ability to adapt;
—*Trusted Controller* represents a designed or given part (may be a given system such as a legacy system but is here represented as a designed problem domain) of the system which provides control functionality;
—*Control Delegator* represents a designed part of a system which is responsible for delegation of control;
—*Monitor* represents a designed part of a system which is responsible for monitoring;
—*Plant* represents the process or equipment to be controlled, defined as a given problem domain;
—*Operating Regions Delegation Scheme* represents the problem of specifying the diversification of the operational state space in a manner which allow specialised controllers to operate in different regions of the operational state space. This is represented as a designed description and is associated with the Control Delegator.

Problem Frame: Figure 4 illustrates and adaptation of the problem frames notation [Jackson 2001] for expressing SACS patterns and specifies the main problem domains related to the division of an operational state space and identifies the need to specify requirements (with reference to the requirement elements in Figure 4) addressing the problems of:

(i) capture the main intent of the pattern by a system level specification of requirements;

(ii) accounting for controller limitations;

(iii) accounting for monitoring and detection limitations;

(iv) accounting for the hazards related to control operations that must be avoided.

The letters on the arcs of Figure 4 denote the interfaces between problem domains. The descriptive names associated with these letters, as provided in the lower frame, represent shared phenomena (e.g. events) which interact between problem domains.

Based on an identification of relevant problem domains, their interfaces and shared phenomenon we may derive the generic requirements which must be addressed when designing a solution. With respect to capturing the pattern intent identified by **problem (i)**, the main aspects to be resolved are:

a) to provide a system solution that optimise service and guarantee safety by the use of a diverse set of controllers;

b) to partition the state space such that for each state of the operational state space, at least one controller (in the set of controllers) may provide service;

c) to partition the state space such that for each state of the operational state space, the system shall always provide a safe service;

d) to delegate control in such a manner that for each state of the operational state space, the adaptable controller is utilised given that safety integrity of the system may not compromised by the adaptive control feature or otherwise do not represent an inappropriate means of control.

The main aspects which influence the state space partitioning with respect to **problem (ii)** are:

a) the ability to delimit AC operations in such a manner that there is still a benefit of having an AC;

b) to assure that suboptimal AC behaviour does not negatively affect safety;

c) to assure that erroneous AC behaviour may be mitigated;

d) the ability of a TC to guarantee safe behaviour;

e) the ability of a TC to mitigate erroneous AC actions.

The main aspects which influence the state space partitioning with respect to **problem (iii)** are:

a) the ability to detect erroneous AC behaviour;

b) the ability to detect and analyse the need to adapt AC in order to meet stated goals;

c) the ability to detect system component events which might have a negative effect on safety;

d) the ability to detect Plant and environment events which might have a negative effect on safety.

The main aspects which influence the state space partitioning with respect to **problem (iv)** are:

a) the hazards associated with the regions of the state space for which adaptable control feature are sought;

b) the failure modes or performance fluctuations that might be associated with the AC;

c) the ability to provide sufficient safety margins in the AC controlled part of the state space such that hazards associated with erroneous AC behaviour may be safely handled.

Requirements: This section specifies the generic requirements and capture the main specification challenges when approaching the problem as defined in the *intent* section. The requirements are derived from the description of the problem as contained in the sections *Problem Context* and *Problem Frame*.

The requirements are contained within a requirements set that contains requirements for each designed domain identified such that Adaptable Controller requirements are denoted with the letters *AC* for short, Trusted Controller requirements are denoted with *TC*, Control Delegator requirements denoted with *CD* and Monitor requirements denoted with *M*. In addition requirements are defined addressing system specific requirements, then denoted with the letter (S) for short. For each requirement there is given a short ID (e.g. AC or TC identifying the respective requirement set) and a running number. In addition a note paragraph giving additional informative guidance is associated with each requirement. The note paragraph also gives reference to the associated problem description and paragraph for which the requirement originated as a means to provide traceability.

The *State Space Subset requirement set* is defined as:

S.1 The *System* shall by the use of a *set of controllers* provide *capability* according to a safety preserving application of control means in *Operating Region*.

NOTE: Refer to problem (i), paragraph a). Here capability may refer to the implementation of the rules for delegation of control in some sort of a delegation scheme, algorithm or other mechanism that may be determined to be safety preserving. The set of controller may be any configuration of controller means, e.g. a set consisting of several different adaptable controllers, conventional programmable controllers and human controllers.

S.2 The *System* shall by appropriate means incorporate a diversification of the *Operating Region* such that a diverse set of controllers may operate in different parts of the operational domain specified by controller specific *Controller Operating Region*

NOTE: Refer to problem (i), paragraph b). If the Operating Region represents the state space for which the system is intended to provide service, then a Controller Operating Region needs to be specified for each controller, specifying the partition of the state space for which it is intended to provide service. As it is assumed that different controllers may not necessarily share a defined Controller Operating Region (depends on the context), it must be assured that there is at least one means of control for each possible state in the Operating Region.

S.3 The *System* shall ensure that at least one *safe means of control* for every state specified by *Operating Region*.

NOTE: Refer to problem (i), paragraph c). In the pattern, TC is assumed to be an inherently safe means of control. The ability of the AC to be a safe means of control depends on the mitigations that may be provided by the system and the delimitation AC control defined by the AC specific Controller Operating Region. Safe means of control may also refer to the ability to use a human operator, any last resort function or protection system in order to assure safe operations or otherwise graceful degradation of service.

S.4 The *System* shall delegate control to the *most suitable controller* with respect to the goal of providing high performance and assure safe control for every state delimited by the specification of *Operating Region*.

NOTE: Refer to problem (i), paragraph d). Several controllers may qualify to provide service in a specific region of the state space (in the event of overlapping Controller Operating Regions). For each operational state, the most suitable controller with respect to the ability to achieve performance goals shall be chosen under the assumption that the sample of controllers to choose from may be safely utilised in the scenario. Potential harmful controllers for a specific operational scenario shall always be suppressed.

AC.1 The *AC* part of the *System* shall/should provide *capability* within *Adaptable Controller Operating Region*.

NOTE: Refer to problem (ii), paragraph a). The capability that are sought needs to be specified together with a description of the assumed trade-offs in order to provide a convincing argument that the adaptive feature will yield *a positive net effect in the associated Adaptable Controller Operating Region. The Adaptable Controller Operating Region represents the specification of the delimitation of AC operations.*

AC.2 Suboptimal *AC* behaviour with negative effect on safety shall be mitigated at the system level or sub-system level by *mitigations*.

NOTE: Refer to problem (ii), paragraph b). Suboptimal behaviour and their potential negative effects on safety needs to be specified such that proper mitigations may be defined. Mitigations here refer to the set of means applied on any system level such that no non-acceptable system failure due to suboptimal AC behaviour may be experienced.

AC.3 Erroneous *AC* behaviour with negative effect on safety shall be mitigated at the system level or sub-system level by *mitigations*.

NOTE: Refer to problem (ii), paragraph c). Erroneous behaviour and their potential negative effects on safety needs to be specified such that proper mitigations may be defined. Mitigations here refer to the set of means applied on any system level such that no non-acceptable system failure due to erroneous AC behaviour may be experienced.

TC.1 The *TC* part of the *System* shall guarantee *capability* in *Trusted Controller Operating Region*.

NOTE: Refer to problem (ii), paragraph d). Capability here refers to functionality that are intended to be provided by the TC. The Trusted Controller Operating Region represents the specification of the part of the state space for which TC is intended to be applied.

TC.2 The *TC* part of the *System* shall guarantee *capability to mitigate* erroneous AC behaviour.

NOTE: Refer to problem (ii), paragraph e). Capability to mitigate here refers to the ability of the TC, in the event of erroneous AC behaviour, to “takeover” control and mitigate negative system effects. The problem may be associated with the challenge of providing Byzantine Fault Tolerance.

M.1 The *M* part of the *System* shall detect erroneous AC behaviour by *capability*.

NOTE: Refer to problem (iii), paragraph a). Capability here refer to the set of means which provide the ability to detect erroneous behaviour of the AC.

M.2 The *M* part of the *System* shall detect and analyse the need to adapt AC by *capability*.

NOTE: Refer to problem (iii), paragraph b). The feature of detection and analysis of the need to adapt may as an alternative be a part of the adaptation logic of the AC. Then the requirement should be defined in the context of the AC Requirement Set.

M.3 The *M* part of the *System* shall by *capability* detect sub-system and component events and failures that may lead to negative effects on safety.

NOTE: Refer to problem (iii), paragraph c). Capability here refers to the sum of monitoring means, e.g. cross-monitoring and self-monitoring mechanisms in the system, and the ability to accumulate information in a monitor part of the system such that the internal health of the system may be determined.

M.4 The *M* part of the *System* shall by *capability* detect Plant and environment effects which may lead to negative effects on safety.

NOTE: Refer to problem (iii), paragraph d). Capability here refers to the sum of monitoring means, e.g. sensor inputs or external monitoring systems, and the accumulation of information to a monitor part of the system such that the health of the system to be controlled (e.g. appliance damage) and its environment (e.g. detect radiation leakage to environment) may be determined.

CD.1 The *CD* part of the *System* shall by *capability* delegate control according to a safety preserving scheme for application of control means.

NOTE: Refer to problem (iv), paragraph a). Here capability may refer to the implementation of the rules for delegation of control in a Delegation Scheme, algorithm or other mechanism that may be determined to be safety preserving.

CD.2 The *CD* part of the *System* shall by *capability* delegate control to appropriate means in the event of hazardous AC performance fluctuations.

NOTE: Refer to problem (iv), paragraph b). Capability here may refer to the implementation of functionality to accommodate unexpected instability, fluctuations or other destabilising behaviour of the AC due to adaptation.

CD.3 The *CD* part of the *System* shall by *capability* incorporate sufficient *safety margins* such that erroneous AC behaviour may not propagate to the system level and invalidate the ability to provide safe service.

NOTE: Refer to problem (iv), paragraph c). Safety margins may here be interpreted to represent the set of means applied in order to provide fault containment. It may not be possible to guarantee that an Adaptable Controller of arbitrary assurance level may not fail in an unpredictable manner, but one may demonstrate that an arbitrary failure to such a component is detected and handled, mitigated or otherwise not possible to be manifested as a system failure.

As can be seen from the specification of requirements, the *State Space Subset* requirements set consists of a total of 16 requirements. The requirements are divided among the sets in the following manner: 4 *System* requirements, 3 *Adaptable Controller* requirements, 2 *Trusted Controller* requirements, 4 *Monitor* requirements and 3 *Control Delegator* requirements.

3.3 Trusted Backup: Design Pattern

Name: Trusted Backup.

Intent: The intent of the pattern is to allow an adaptable controller, that might be of an arbitrary assurance level, to control a safety related plant. The adaptable controller is here introduced in order to accommodate changes in the plant as a means to uphold service despite changes (e.g. adjust to plant degradation) or improve service over time (e.g. better fitting of control law). The intent is to divide the operational state space associated with some controlled plant into regions. The adaptable controller is in this control scheme not a trusted component thus it is only granted control privileges in those regions (the recoverable states), called safety regions, where there are no negative effects of controller failure. In order to provide service in those regions which are not safety regions or take over control if the adaptable controller experiences failure, a redundant trusted controller is granted control privileges. The controllers are developed according to different principles in which a *Trusted Controller* guarantees safe control but is less resilient to changes in the plant whereas an *Adaptable Controller* accommodate changes and can optimise accordingly but does not guarantee safe control. The *System* controls the plant in a manner which emphasizes both optimal and safe control by delegating control privileges to the most suitable controller depending on the state of the plant.

Motivation: The pattern employ a strategy where the potential benefit of utilising adaptable means of control is provided, but where effort required to demonstrate that the Adaptable Controller is safe is to a large degree evaded. The Adaptable Controller is intended to provide an increased level of performance compared to a conventional software system. Any erroneous operation of the Adaptable Controller shall not negatively impact safety thus the Adaptable Controller may be of an arbitrary safety assurance level.

An example of such a problem may be an adaptable flight control system where adaptiveness is used to improve flight performance of an aircraft. We assume here that adaptive behaviour may be introduced in order to improve roll axis rotation control with the effect of improving passenger comfort during flight. With respect to passenger comfort, we further assume that the optimal roll angle should be such that the plane is stabilised. When the plane is in a reasonably stable state, an erroneous actuation of control means affecting the roll angle negatively with respect to the optimal comfort level will not have any immediate negative effect on safety, it will have an effect on the flight controller's ability to provide optimal comfort level for its passengers. Given that the adaptable controller applies its actuators in such a manner that the boundary with respect to roll rotation angle is reached, the adaptable controller may be deactivated and the conventional autopilot or the pilot may take over control.

Applicability: The Trusted Backup Design Pattern is suitable to apply in the following situations:

—in control scenarios where there is a need for a high degree of exploration of alternative controller solutions, e.g. prototyping or upgrade scenarios, as the safety of the system is not ensured by the adaptable controller component and thus the adaptation logic and the control logic may be rapidly changed;

—when there is an existing trusted system. Adaptable control functionality may then be provided by adding the Adaptable Controller, , Control Delegator and the Monitor (if not such a component is already present).

Related Patterns: The *Trusted Backup* Design Pattern is related to the other patterns in the following manner:

satisfies the requirements associated with the *State Space Subset* Requirement Pattern.

demonstrates relationship from the *Adapting Within a Constrained State Space* Safety Case Pattern as instantiation of the design pattern should provide a suitable specification of a system with respect to the main lines of the safety demonstration.

Structure: Figure 5 illustrates the main structure of parts and their interfaces, annotated in UML notation [Object Management Group (OMG) 2010], that play a part in obtaining a *Trusted Backup* design.

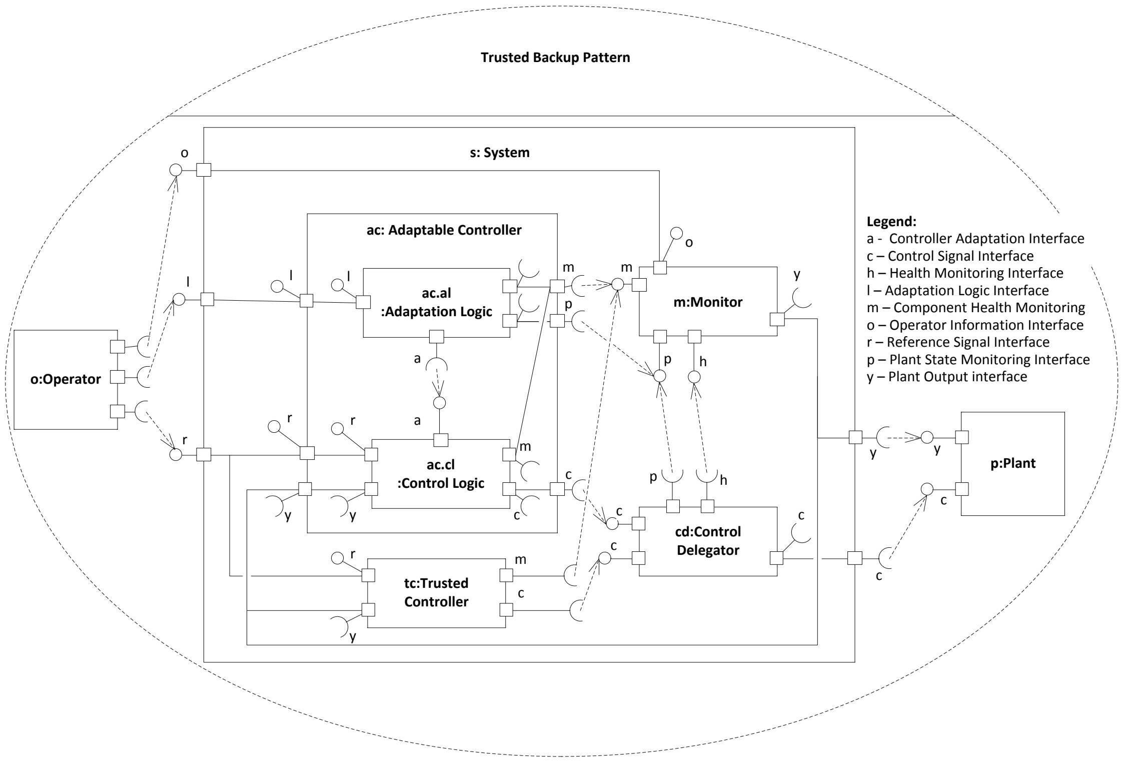
Trusted Backup Pattern
s: System
ac: Adaptable Controller
ac.al :Adaptation Logic
ac.cl :Control Logic
tc:Trusted Controller
m:Monitor
cd:Control Delegator
o:Operator
p:Plant
Legend:
a - Controller Adaptation Interface
c – Control Signal Interface
h – Health Monitoring Interface
l – Adaptation Logic Interface
m – Component Health Monitoring
o – Operator Information Interface
r – Reference Signal Interface
p – Plant State Monitoring Interface
y – Plant Output interface

Fig. 5. Structure

Participants: The following participants and their respective responsibilities are represented in the *Trusted Backup* pattern:

—*System* represents the overall system, consisting of parts, and that interacts with a *Plant* and some *Operator*.

—*Operator* represents some human operator that interacts with the system.

—*Plant* represents the process or equipment to be controlled. The *Adaptable Controller* is here decomposed into an *Adaptation Logic* part and a *Control Logic* part.

—*Control Logic* is responsible for providing control signals based upon the current plant state and reference data points. Adaptation of the *Control Logic* is here provided by an external adaptation mechanism (external to the respective controller) identified as *Adaptation Logic*.

—*Adaptation Logic* is responsible for detecting and accommodating changes to the plant or the environment in which the plant operates in order to uphold or improve performance. It is here assumed that the absence of side-effects due to adaptation may not be adequately evaluated thus a high level of trust may not be awarded the Adaptable Controller's Control Logic nor the Adaptation Logic.

—*Trusted Controller* is responsible for providing control signals based upon the current plant state and reference data points. It is assumed that it may be demonstrated that the Trusted Controller always operate in a correct and sufficiently safe manner.

—*Control Delegator* is responsible for delegation of control such that control privileges is granted to the most suitable controller of a set of alternative controllers in such a manner that plant safety is always assured and secondly that the most optimal controller for a given situation is always sought. All controllers operate in parallel where a set of delegation rules (captured in the Delegation Scheme description of the Control Delegator) govern which controller is granted control privileges in each state, the remaining controllers being suppressed.

—*Monitor* is responsible for monitoring the state of the plant, monitoring the state of the system, and provides information on status to relevant system parts.

Collaborations: Delegation of control is governed by the *Control Delegator* part which based on received/requested *System* and *Plant* state information delegate control to one of its delegates. Given that there are no detected failures to any of its delegates, the *Control Delegator* delegates control according to a defined *Delegation Scheme*.

Controller adaptation is initiated in response to the result of the detection and analysis of change functionality provided by the *Adaptation Logic*. Once the need to adapt is identified, the *Adaptation Logic* adapts the *Control Logic* accordingly. In order to assure a safe control process, the Adaptable Controller is deactivated from providing control operations in the time period when it is modified. The *Trusted Controller* will then continuously provide service and thus safety is assured.

Known Uses: Åström and Wittenmark [1994] reference a number of applications with adaptive control. One application is related to a chemical reactor control system where an adaptive controller was implemented on a reactor at Berol Kemi AB in 1982. As poor control may not only result in lowered production but may led to potentially fatal events as explosions, to uphold the safety property of the plant is vital. The system operator may switch between a conventional controller with manual mode tuning or an adaptive controller with automatic tuning to control the flow and temperature of the chemicals in the plant. Experiences showed that that temperature fluctuation was significantly reduced with the adaptive system, in addition the operator could focus on other tasks as time spent supervising reactor temperature was not necessary.

A representative example is described in [Schumann et al. 2006]. The authors refer to a NASA project called IFCS (Intelligent Flight Control System). The IFCS project utilised an on-line adaptive neural network in order to optimize aircraft performance during normal and adverse conditions. The neuro-controller was designed to enable a pilot to maintain control and safely land an aircraft that had suffered a major systems failure or combat damage. Control surface failures may conflict with the design assumptions of an aircraft flight control system, with the effect that it is unable to handle the situation. The IFCS neuro-controller compensate for discrepancies between a reference model of the flight dynamics to any "new" flight dynamics model in order to maintain the best possible flight performance. The adaptive neural network software "learns" the new flight characteristics, on-board and in real time, thereby helping the pilot to maintain or regain control and prevent a potentially catastrophic aircraft accident.

The *Trusted Backup* pattern enable utilisation of adaptive control in a manner similar to the principles advocated in the Simplex Architecture [Sha 2001]. Sha [Sha 2001] refer to the Boeing 777 flight control system as an example of a system that uses the ideas of the Simplex Architecture in practise. The system uses two controllers, a sophisticated “normal” controller which is customised for the Boeing 777 and a secondary controller which is simple and well proven through 25 years of use and based on the Boeing 747 control laws. The higher level of performance is provided by the “normal” controller and it must operate within the bounds given by the secondary controller which assure safe flight control. The Boeing 777 case is comparable to the Trusted Backup pattern main line of though. Although an adaptable controller is not utilised in the Boeing 777, the “normal” controller represent a software controller that is an assessment and safety demonstration challenge. The secondary controller and the mechanism for switching among controllers is a means for providing safety assurance.

Safety Features: An important part with respect to safety assurance in the pattern is provided by the *Control Delegator*. It must be assured that it grants control privileges to the most suitable controller at all times. The pattern is suitable to apply when there are no means of reverting to a previously stored plant state given a failure scenario, thus it would be important to support a forward recovery strategy. Given that an erroneous control signal is given by the Adaptable Controller, the potential danger this signal inflicts must be mitigated from the current execution point. A means for achieving a forward recovery feature is provided here by redundant controllers, health monitoring and conservative control delegation rules. Potentially hazardous control behaviour may arise if the Control Delegator grants control privileges to a non-trusted control component in a plant state where a worst case erroneous control sequence may not be mitigated in successive control loops by the Trusted Controller. With respect to providing guarantee for safe delegation of control, the following must be known: (1) the potential hazards associated with the plant; (2) worst case failure scenario and the failure modes of the adaptive controller which may lead to hazardous control and (3) the capability of the Trusted Controller to stabilise the Plant.

Adaptability: Software adaptations are intended to be limited to the Adaptable Controller only. The objective of the Adaptable Controller is to optimise control with respect to some performance goal, how this is achieved is of minute concern although a component named *Adaptation Logic* is assumed to contain the rules for how to adapt the controller according to some identified need to adapt. Detection of changes in the plant and its environment is expected to be performed by the *Adaptation Logic* on the basis of data provided by the *Monitor*.

Variability: Variability of the system is isolated to the *Adaptable Controller* part, more specific, the *Control Logic*. The *Control Logic* part might exist in various versions over time as the *Adaptation Logic* modifies the control function in response to plant or environmental events. A specification of the potential for system variability thus may be provided once the adaptation logic is specified.

There will be no variability of the system service with respect to the region of the operating state space for which only the Trusted Controller may operate as the Trusted Controller is assumed to be a deterministic controller. Variability of the system service will be experienced in the region of the operating state space for which the *Adaptable Controller* may operate and in those scenarios it is granted control privileges as different version of the *Control Logic* may provide different output when acting on the same input.

3.4 Adapting Within a Constrained State Space: Safety Case Pattern

Name: Adapting Within a Constrained State Space

Intent: The intent is to provide an argumentation structure for how to demonstrate that an adaptable control system is safe when the Adaptable Controller part of the system is: (1) adapted by an unconstrained adaptation mechanism; (2) limited to operate in a part of the total operating state space for which there is sufficiently large safety margins to compensate for erroneous behaviour; (3) granted control privileges by a mechanism for delegating control according to a safe delegation scheme; and (4) backed up by a Trusted Controller which is guaranteed to provide safe behaviour in the total operating state space.

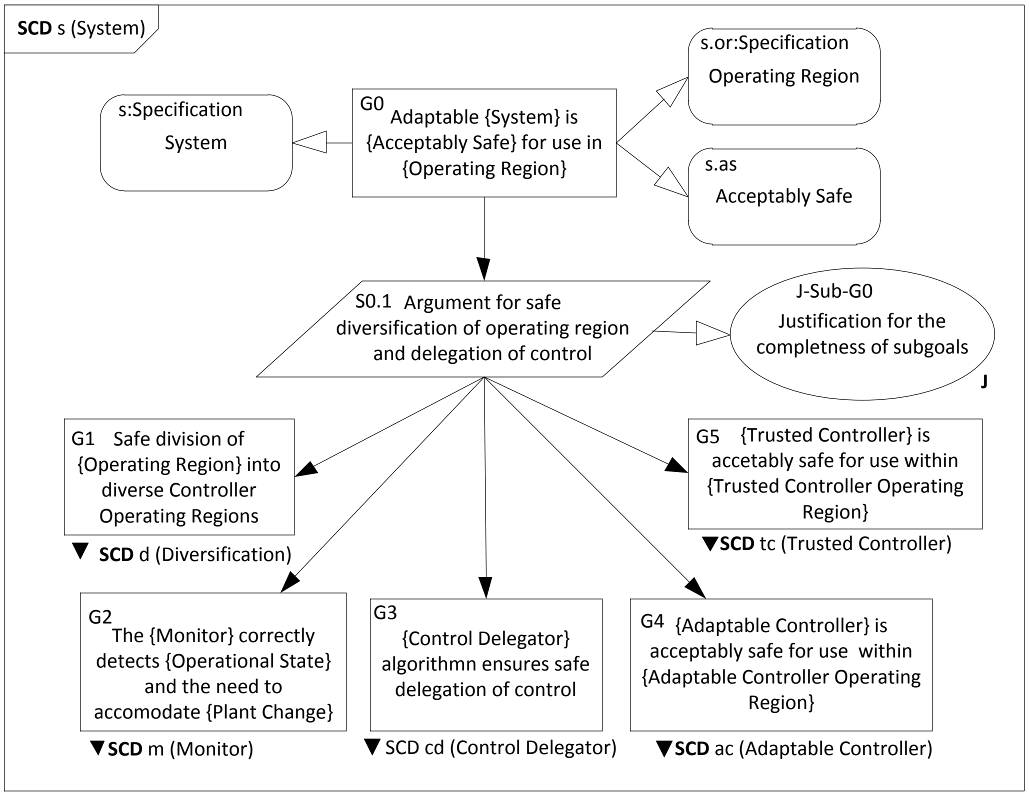
SCD s (System)
s:Specification
System
G0 Adaptable {System} is {Acceptably Safe} for use in {Operating Region}
s.or:Specification
Operating Region
s.as
Acceptably Safe
S0.1 Argument for safe diversification of operating region and delegation of control
J-Sub-G0
Justification for the completness of subgoals
J
G1 Safe division of {Operating Region} into diverse Controller Operating Regions
▼ SCD d (Diversification)
G5 {Trusted Controller} is accetably safe for use within {Trusted Controller Operating Region}
▼SCD tc (Trusted Controller)
G2 The {Monitor} correctly detects {Operational State} and the need to accomodate {Plant Change}
▼SCD m (Monitor)
G3 {Control Delegator} algorithmn ensures safe delegation of control
▼ SCD cd (Control Delegator)
G4 {Adaptable Controller} is acceptably safe for use within {Adaptable Controller Operating Region}
▼SCD ac (Adaptable Controller)

Fig. 6. Main Safety Argument

Motivation: The motivation is to allow a freedom in how an Adaptable Controller is adapted by not assigning any trust to this part of the of the system but rely on other means in order to assure safety. Whilst the Adaptable Controller is assumed to provide an increase in performance in a restricted part of operation domain, safety is assured by a Trusted Controller, a Control Delegator and a Monitor mechanism.

Applicability: The *Adapting Within a Constrained State Space* pattern is suitable to apply in combination with systems that:

—makes use of monitoring as a means to detect plant states the state of its environment, and to manage information on system component failure;

—makes use of a redundant set of controllers where adaptable behaviour is applied to a subset of the controllers;

—is associated with an operational state space that may divided such that adaptable control may be limited to a safe subset of this state space.

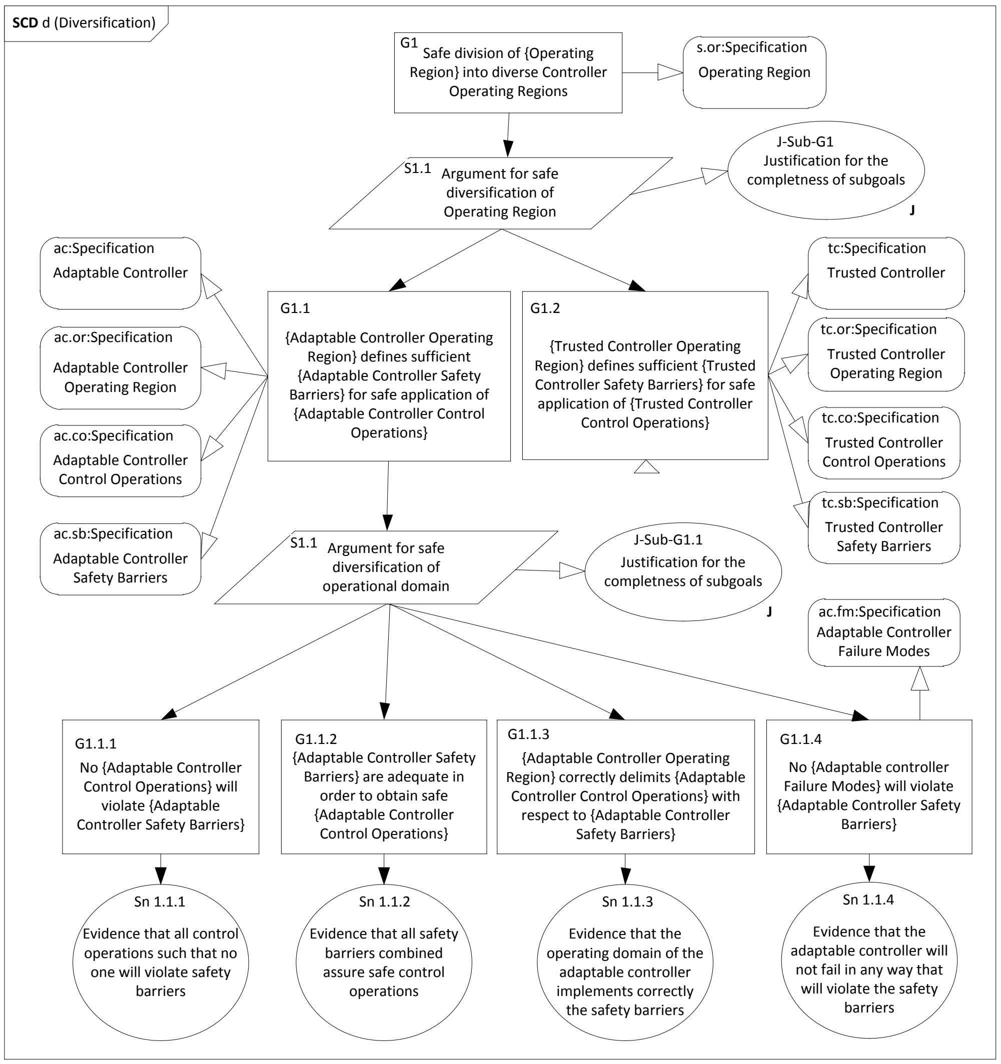
SCD d (Diversification)
G1 Safe division of {Operating Region} into diverse Controller Operating Regions
s.or:Specification Operating Region
S1.1 Argument for safe diversification of Operating Region
J-Sub-G1 Justification for the completness of subgoals
J
ac:Specification Adaptable Controller
ac.or:Specification Adaptable Controller Operating Region
ac.co:Specification Adaptable Controller Control Operations
ac.sb:Specification Adaptable Controller Safety Barriers
G1.1 {Adaptable Controller Operating Region} defines sufficient {Adaptable Controller Safety Barriers} for safe application of {Adaptable Controller Control Operations}
G1.2 {Trusted Controller Operating Region} defines sufficient {Trusted Controller Safety Barriers} for safe application of {Trusted Controller Control Operations}
tc:Specification Trusted Controller
tc.or:Specification Trusted Controller Operating Region
tc.co:Specification Trusted Controller Control Operations
tc.sb:Specification Trusted Controller Safety Barriers
S1.1 Argument for safe diversification of operational domain
J-Sub-G1.1 Justification for the completness of subgoals
J
ac.fm:Specification Adaptable Controller Failure Modes
G1.1.1 No {Adaptable Controller Control Operations} will violate {Adaptable Controller Safety Barriers}
G1.1.2 {Adaptable Controller Safety Barriers} are adequate in order to obtain safe {Adaptable Controller Control Operations}
G1.1.3 {Adaptable Controller Operating Region} correctly delimits {Adaptable Controller Control Operations} with respect to {Adaptable Controller Safety Barriers}
G1.1.4 No {Adaptable controller Failure Modes} will violate {Adaptable Controller Safety Barriers}
Sn 1.1.1 Evidence that all control operations such that no one will violate safety barriers
Sn 1.1.2 Evidence that all safety barriers combined assure safe control operations
Sn 1.1.3 Evidence that the operating domain of the adaptable controller implements correctly the safety barriers
Sn 1.1.4 Evidence that the adaptable controller will not fail in any way that will violate the safety barriers

Fig. 7. Diversification of Operating Domain Argument

Related Patterns: *Adapting Within a Constrained State Space* safety case pattern is related to other patterns in the following manner:

demonstrates by its safety demonstration system designs based on the principles given by the *Trusted Backup* Design Pattern

Argument Structure: Figure 6 illustrates the top level argument specified in an adapted version of the GSN [Kelly and Weaver 2004] notation in order to express SACS patterns and provides references to a set of diagrams which jointly represent a decomposition of the argument. Due to space restrictions, only the part of the decomposition related to diversification of the Operating Region, represented by Figure 7, is provided.
It assumed that a *System Specification* is provided or developed that contains and sufficiently specify information that may be associated with the contextual elements of Figure 6 and Figure 7. Each named contextual element in the safety

case diagrams (e.g. s - System, s.or - Operating Region and s.as - Acceptably Safe of Figure 6) is thus expected to be described by or referred from the system specification.

Safe Adaptation: The main claims which is argued by this safety case pattern with respect to safe adaptation rely on the assumptions that:

—the objective for introducing adaptability is related to a demand for increasing performance, thus the main goal of the argument is to demonstrate that there are no negative effects of the adaptive feature;

—the adaptive controller operate in a constrained part of a larger operational state space where there exists safety margins such that if the adaptive controller fails to operate correctly, the potential negative effects on operations may be mitigated by a backup controller.

4. USE OF THE LANGUAGE

The use of the SACS pattern language will be explained with respect to how one may apply the *Extended Trusted Backup* composite pattern, described in Section 3, on a railway application introduced in Section 4.1.

4.1 Railway Interlock System Example

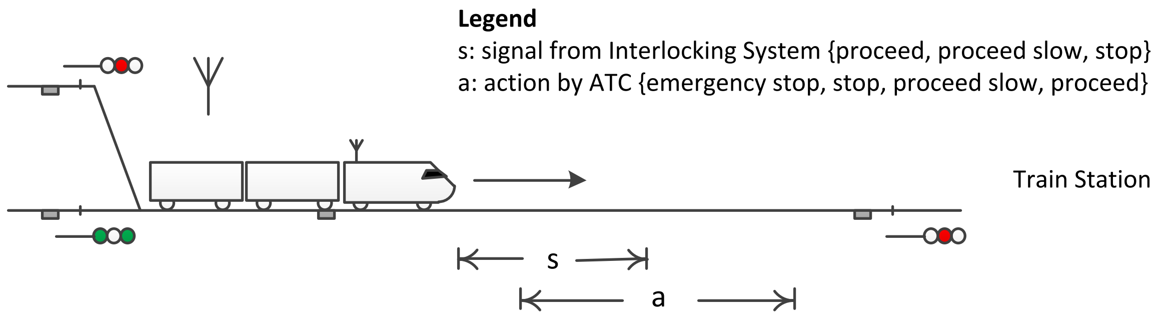
Legend
s: signal from Interlocking System {proceed, proceed slow, stop}
a: action by ATC {emergency stop, stop, proceed slow, proceed}
Train Station
s
a

Fig. 8. Railway Interlocking Adaptive Control Example

A railway interlock system controls a set of signal apparatus and appliances in order to prevent conflicting train movements. A typical interlock system will in the situation illustrated in Figure 8, detect that a train occupy a section of the track and signal proceed if the destined track section at the train station is free and that all other conditions associated with this train movement are satisfied.

Given that the correct position of a train may be obtained, e.g. by an ERTMS level 2 compliant set of appliances [Winter 2009; Stanley 2011], as opposed to the more rough track section detection, it is assumed that a more optimal traffic flow is obtained if the proceed signal is given as late as possible. In this scenario the signal will instead of indicating proceed as soon as the conditions for this are satisfied, continue to signal stop (stop is the default signal) until a point where the train should either: (1) slowdown in order to stop before a designated stop point; or (2) proceed and thus avoid a slowdown and loss of kinetic energy. The difference between the points in time when the train could at its earliest be given proceed, and the point in time when a signal is given according to a optimally late signalling strategy, may be allocated for other interlock operations. The system will then not reserve the designated train station track section for the train until it is absolutely necessary, thus the very same piece of the track is available for other operations. If we assume that the interlock system has planned many operations at the train station at approximately the same time, a smart system is able to optimise execution of planned train movements at the station while our train is approaching and thus an opportunity to exploit available capacity.

4.2 Applying The Composite Pattern

We will detail how the composite pattern illustrated in Figure 3 may be applied with respect to the railway interlock example by elaborating upon the application of its constituent patterns.

The **Requirement Pattern** named *State Space Subset* detailed in Section 3.2 may be instantiated as follows with respect to the interlocking example.

We may associate the problem domains specified in Figure 4 such that the *Plant* represents the signalling appliances. The system functionality which we want to derive requirements for is identified by the decomposed machine illustrated as the dotted machine element which encapsulates the designed domains *Adaptable Controller*, *Trusted Controller*, etc. In the railway scenario, problem domains such as the *Adaptable Controller* may represent an adaptive interlock system and the *Trusted Controller* may represent a standard interlock system. Although all these problem domains may be parts of a larger system we are developing, the functionality which they represent are explicitly identified as problem domains. The problem of specifying the *Operating Regions* is in the railway context represented by the problem of diversifying the set of states for which the interlock system shall provide operations. If we assume one Adaptable Interlock Controller (or system) and one Trusted Interlock Controller, the Trusted Interlock Controller may be granted control privileges in any operational state for which the total system shall provide interlock operations, whereas the Adaptable Interlock Controller may be granted control privileges only when a train is approaching the train station and when the conditions for giving "proceed" are satisfied.

The main challenge when applying the pattern is to assure a safe strategy for delegation of control by specifying operating regions and constraints in accordance with the capability for monitoring the current state and the capability of the different controllers to enforce safe control within their dedicated operating regions. The pattern assumes that a minimum level of performance and safe operations are provided by a Trusted Interlock Controller. In our scenario this may be by employing a track detection signalling scheme such that if the conditions for giving a proceed signal are satisfied, this is given once the train is detected on the approaching track section. An Adaptive Interlock Controller might use sophisticated algorithms in order to optimise train traffic control without negatively affecting safety as:

—the Adaptable Interlock Controller provide an optimisation effect (yields the possibility to increase the number of train station movements) in this limited part of the operational state space;

—default signal is "stop" which is a safety preserving signal as the train must stop if "proceed" is not given. Delaying the "proceed" signal then enforce an even more conservative signalling regime with respect to safety than the Trusted Interlock Controller;

—given that the Adaptable Interlock Controller provides a non-optimal estimate, in the sense signals "proceed" earlier than optimal, the effect is less time to optimise train station movements. There are no negative safety effects, only negative effect on optimisation;

—given that the Adaptable Interlock Controller provides a non-optimal estimate, in the sense signals "proceed" later than optimal, then there might be negative effects. The Monitor part of the System should detect that the train has moved to a point where emergency braking is the remaining option. In such a scenario, conditions for delegation of control should be specified such that the Trusted Interlock Controller is granted control privileges. Any Adaptable Interlock Controller issued signal is suppressed as the train moves beyond a point where the train must be stopped immediately if the conditions for approaching the destined track section are changed (e.g. destined track is occupied).

If we assume that all the states that are intended to be controlled by the Interlock System represents one Operating Region and that this region is governed by the Trusted Interlock Controller, then the problem of specifying regions is mainly to specify the Adaptive Interlock Controller Operating Region. The premises for granting control privileges to the Adaptive Interlock Controller may be when:

—a train is detected on the approaching track section;

—conditions for giving the proceed signal is satisfied (this could be verified by reading the Trusted Interlock Controller signal, which should be "proceed" but which is in this control scenario is suppressed);

—the train has not moved beyond a point on the track where emergency braking is the only option in order to stop the train before a designated stop point.

In our example the objective of the Adaptable Interlock Controller is twofold. The intention is to give proceed signal as late as possible in order to: (i) increase time for performing other train station interlock movements; and (ii) signal sufficiently early in order to avoid a slowdown of the train by braking and thus preserve kinetic energy and provide energy efficient control. The optimal point on the track to give a "proceed" signal is expected to be at sufficient distance to the point where immediate braking must be considered. Thus the unwanted effect that the Adaptable Interlock Controller pushes the optimal point towards this boundary, providing control operations where trains may experience emergency braking, has a sufficient low probability. Although the emergency braking preserves the safety of the train and the infrastructure, it is an unwanted use of control means intended for emergency use only.

The **Design Pattern** named *Trusted Backup* detailed in Section 3.3 may be instantiated as follows with respect to the interlock example.

If we assume that the *Trusted controller* illustrated in Figure 5 can be considered sufficiently safe for the intended control purpose (e.g. a formally verified conventional programmed logic based Trusted Interlock Controller), then safe control operations should be guaranteed by this part alone. The *Monitor* part provides, as in any current interlock system, information on the state of the different appliances involved in the control scenario to the controller. The increased system complexity, compared to a conventional control systems, is provided by the *Control Delegator* part which is responsible for delegation of control, and the *Adaptable Interlock Controller*.

In order to assure safe delegation of control, the *Control Delegator* might be implemented by some programmed logic as a set of formally verifiable rules according to the principles outlined with respect to application of the requirement pattern. As we argued that the outlined delegation scheme would be safety preserving, we will focus on the functionality for optimising the signalling behaviour.

One alternative implementation of the *Control Logic* part of the *Adaptive Interlock Controller* may be by a control law represented as a mathematical formula that considers several parameters (e.g. train type, load, speed) in order to estimate an optimal late signalling point. The formula may be obtained by the *Adaptation Logic* part by means of combining statistics over past events (e.g. effect of different weather conditions on braking time), data on the different braking patterns of trains under different load conditions, etc. in order to update the control law as more data is obtained or better algorithms are developed. The complexity of the algorithms used to obtain the control law or the complexity of the control law does not provide an assessment problem, the result is still a controller which issues either a "proceed" or a "stop" signal. The scenarios in which these commands may be issued are handled with sufficient determinism by the *Control Delegator* part, thus the *Adaptable Interlock Controller* may be of arbitrary assurance level. It is of minute concern how often the control law is updated as safe control is sustained by the *Trusted Interlock Controller* during the update.

The **Safety Case Pattern** named *Adapting Within a Constrained State Space* detailed in Section 3.4 may be instantiated as follows with respect to the interlock example.

The intent of the Safety Case Pattern is to provide a generic argument structure in order to outline how claims, arguments and evidences may be structured systematically in order to be able to demonstrate that a particular type of system is sufficiently safe. In the context of our example as provided in Section 4.1, this means to demonstrate that the overall interlock system is safe.

When instantiating the pattern, the context entities as represented by e.g. *s - System*, *s.or - Operating Region* and *s.as - Acceptably Safe* of Figure 6 (context entities in GSN notation) must be given a real meaning. With respect to our railway example, the *System* may be represented by the overall interlock system. The *Operating Region* may be represented by a specification of the railway infrastructure and the environment and the states for which the interlock system shall operate. What is *Acceptably Safe* at the system level may be provided by applicable laws, regulations and standards.

The goals G1 to G5 are detailed in separate diagrams (not provided here due to space considerations), and if fulfilled support the overall claim G0 that the system is acceptably safe. The justification for the completeness of G0 by the goals G1 to G5 should be specified in J-Sub-G0. An outline of a possible instantiation of the pattern with respect to the main goals may be as follows:

G1 The *Operating Domain* represents the set of all states for which interlock operations should be given. We argued earlier that the operational state space for which the adaptable controller may be allowed to operate could be specified with a small set of rules. The G1 goal would in our context be concerned with demonstrating that the set of rules provides an adequate partitioning of the state space.

G2 The goal is concerned with demonstrating that a part of the system correctly identifies or detects the *operational state* at any given time.

G3 The goal is concerned with demonstrating the correct enforcement of the rules for delegation of control. This means to demonstrate that the adaptive controller is only delegated control privileges in the scenarios developed in G1 and that failure events which may invalidate the design intent are detected and handled or otherwise are absent or acceptable.

G4 The goal is concerned with demonstrating that any action performed by the Adaptable Controller is acceptably safe. We stated that obtaining a non-optimal control law might give the proceed signal earlier than intended, later than intended or otherwise at a point where the train must employ emergency break down if proceed is not given. All these scenarios do not affect safety but rather performance if the mitigations proposed may be implemented. Potential failure scenarios arising in the adaptable controller may be acceptable if proper mitigation is provided by other components such that the failures are not allowed to propagate to the system level.

G5 The goal is concerned with demonstrating that the Trusted Interlock Controller, which is represented by a conventional interlock controller, is sufficiently safe and may mitigate Adaptable Interlock Controller failure. In our context this means to emphasise the point that if the Adaptable Interlock Controller erroneously late (with respect to optimisation) delay giving a “proceed”, then “stop” is continuously signalled. This is hardly hazardous but more conservative with respect to safety. If the inability of control is meant sending erroneously early (with respect to optimisation) a “proceed” signal then this is in harmony with the Trusted Interlock Controller. If it is meant sending another signal than “proceed” or “stop” then this should be easily detected by the monitor. If there are some circumstance which alter the conditions such that a “stop” signal should be given, then the premises for granting control privileges to the Adaptable Interlock Controller are not satisfied.

5. CONCLUSION

The patterns presented in this paper are intended to support the design phase of the development life cycle. Typical users are system architects, software designers, and safety engineers. They may use the patterns as a means to effectively outline a conceptual model of a system as well as outlining the safety demonstration challenges.

The SACS pattern language facilitates a complementary viewpoint to the problem of deriving safe adaptive software. The complementary viewpoint is given by the possibility to integrate a set of patterns, from the set of different pattern types, into a composition such that the requirements that may be associated with a particular problem, design solutions for the problem and safety demonstration aspects may be conveniently explored. The SACS approach provides a more nuanced means for evaluating the suitability of a particular design than the GoF [Gamma et al. 1995] style of expressing patterns. The difference is that SACS does not only present a design pattern solution alone, detailed guidance are provided on the issues of requirements elicitation and safety demonstrations. When developing safety critical or safety related systems, the increase in cost compared to systems not important to safety is to a large degree related to the need to demonstrate that these systems are adequately safe. By coupling patterns dedicated to requirements elicitation, design specification and safety demonstration, a nuanced pattern approach is provided. In SACS, a user is given a means to specify the use of a set of patterns by the *Composite Pattern* type. The notation for defining compositions increases the ability to systematically capture and communicate a complex pattern solution. This facilitates a common "language" for expressing the application of patterns and thus ease communication amongst different users about aspects of a specific pattern approach. Compared to the [Alexander et al. 1977] style for combining patterns, the SACS approach is more structured and more visually apprehensible in that a set of graphical entities and a set of rules for their use are given in order to express a specific intent in the composition of patterns.

Future work includes expanding the set of basic patterns and detailing the syntax of the pattern language. The intention is to expand the set of patterns such that at least the alternative strategies provided in the introduction (those introduced but not described in this article) are covered. One of these alternative strategies was to constrain the adaptation mechanism and the other alternative was to incorporate a runtime evaluation mechanism which constrains the committing of modifications to only safe modifications. Future work also includes evaluation of the SACS language through large case studies and other suitable means.

Acknowledgment

This work has been conducted and funded within the OECD Halden Reactor Project, a nuclear research project managed by Institute for energy technology (IFE), Halden, Norway.

Many thanks to our PLOP shepherd Jason Yip and the PLOP Workshop group consisting of: Robert Hanmer, Alexander Nowak, Rick Rodin, Youngsu Son, Jiwon Kim, Tanya Crenshaw and Hiero for their constructive and valuable feedback. Thanks also to Pedro Monteiro for the valuable feedback given at PLOP.

REFERENCES

ALEXANDER, C., ISHIKAWA, S., AND SILVERSTEIN, M. 1977. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York.

ÅSTRÖM, K. J. AND WITTENMARK, B. 1994. *Adaptive Control.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

GAMMA, E., HELM, R., JOHNSON, R. E., AND VLISSIDES, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA.

JACKSON, M. 2001. *Problem Frames: Analysing and Structuring Software Development Problems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

KELLY, T. AND WEAVER, R. 2004. The Goal Structuring Notation - A Safety Argument Notation. In *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases.*

OBJECT MANAGEMENT GROUP (OMG). 2010. Unified Modeling Language Specification. Version 2.3.

SCHUMANN, J., GUPTA, P., AND JACKLIN, S. 2006. Toward Verification and Validation of Adaptive Aircraft Controllers. In *IEEE Aerospace Conference.* IEEE Press, Piscataway, NJ, USA.

SCHUMANN, J. AND LIU, Y. E. 2010. *Applications of Neural Networks in High Assurance Systems.* Springer Verlag, Berlin.

SHA, L. 2001. Using Simplicity to Control Complexity. *IEEE Software 18*, 20–28.

STANLEY, P., Ed. 2011. *ETCS for Engineers.* DVV Media Group GmnH, Eurail Press, Hamburg, Germany.

WINTER, P., Ed. 2009. *Compendium on ERTMS — European Rail Traffic Management System.* DVV Media Group GmnH, Eurail Press, Hamburg, Germany.

PLoP'11, October 21-23, Portland, Oregon, USA. Copyright 2011 is held by the author(s). ACM 978-1-4503-1283-7