A Model-Driven Engineering Approach to Support the Verification of Compliance to Safety Standards

Rajwinder Kaur Panesar-Walawege, Mehrdad Sabetzadeh, Lionel Briand
Simula Research Laboratory, Lysaker, Norway
University of Oslo, Norway
Email: {rpanesar,mehrdad,briand}@simula.no

***Abstract*—Certification of safety-critical systems according to well-recognised standards is the norm in many industries where the failure of such systems can harm people or the environment. Certification bodies examine such systems, based on evidence that the system suppliers provide, to ensure that the relevant safety risks have been sufficiently mitigated. The evidence is aimed at satisfying the requirements of the standards used for certification, and naturally a key prerequisite for effective collection of evidence is that the supplier be aware of these requirements and the evidence they require. This often proves to be a very challenging task because of the sheer size of the standards and the fact that the textual standards are amenable to subjective interpretation. In this paper, we propose an approach based on UML profiles and model-driven engineering. It addresses not only the above challenge but also enables the automated verification of compliance to standards based on evidence. Specifically, a profile is created, based on a conceptual model of a given standard, which provides a succinct and explicit interpretation of the underlying standard. The profile is augmented with constraints that help system suppliers with establishing a relationship between the concepts in the safety standard of interest and the concepts in the application domain. This in turn enables suppliers to demonstrate how their system development artifacts achieve compliance to the standard. We illustrate our approach by showing how the concepts in the domain of sub-sea control systems can be aligned with the evidence requirements in the IEC61508 standard, which is one of the most commonly used certification standard for control systems.**

***Keywords*-UML; Profile; safety; certification;**

I. Introduction

Safety-critical systems are often subject to a stringent safety certification process, aimed at providing assurance that a system is deemed safe by a certification body. Increasingly, system suppliers are asked by such bodies to provide their justification for the safe operation of a system in the form of a *safety case*, which provides well-reasoned arguments based on the collected safety evidence that the overall safety objectives of a system are being met [1]. One approach to certification is to demonstrate, based on evidence, compliance to a safety standard, such as IEC61508 for Programmable Electronic Systems (PES) [2].

Verifying compliance to safety standards proves to be a very challenging task because of their sheer size and the fact that they are mostly textual and subject to subjective interpretation. On the supplier side, they run the risk of missing critical details that need to be recorded *during* system development. This means that they will have to reconstruct the missing evidence after the fact. Doing so is often very expensive, and the outcomes might be far from satisfactory. On the certifier side, poorly structured and incomplete evidence often leads to significant delays and loss of productivity, and further may not allow the certifier to develop enough trust in the system that needs to be certified. It is therefore very important to devise a systematic methodology, which is amenable to effective automated support, to specify, manage, and analyze the safety evidence used to demonstrate compliance to standards.

Motivated by the above challenges, we have studied in our previous work different facets of the problem of safety evidence specification and management. Specifically, we proposed an approach to specify safety evidence using conceptual modeling [3] and a technique for tailoring generic evidence requirements according to sector-specific needs (e.g., in the railways, avionics, and maritime and energy sectors) [4]. A recurring theme in this earlier work is the use of standard Model-Driven Engineering (MDE) technologies, such as UML [5] and OCL [6] for specification, storage, and analysis of safety evidence information.

In this paper, drawing on the same MDE principles underlying our previous work, we develop a novel approach for assisting system designers in relating the concepts of their application domain to the evidence requirements of the standards that apply to the domain. The research is motivated by a natural and real need that we have observed in software safety certification. Specifically, the majority of the evidence artifacts that the suppliers record are based on the supplier's concepts for the application domain, as opposed to the concepts of the certification standards. The absence of an explicit and precise link between the two conceptual frameworks can pose two main challenges. First, the certifier may not be able to comprehend the evidence, and second, it becomes very difficult to verify whether the evidence collected using the domain concepts is covering all the evidence aspects mandated by the standard.

To give a concrete example, in the IEC61508 standard, a Programmable Electronic System (PES) is the system for controlling or monitoring one or more programmable electronic devices, including all elements of the system such as sensors, communication paths, and actuators. It has software that is used to send commands for controlling

the various different types of equipment. A sub-sea control system on the other hand, is made up of a Sub-sea Control Module (SCM) that incorporates a Sub-sea Electronics Module(SEM). The SCM executes the commands for opening or closing valves that control the oil well. These commands are sent from the Sub-sea Control Unit (SCU) which is software that run on the oil rig in what is called the Topside Processing Unit(TPU).[7], [8]. In this scenario, the certifier needs to know which is the PES, and which is the software system. The PES in this case is the SCM and the software controlling and monitoring it, is the SCU. The correlation of these simple pieces of information provides clarification to the certifier who needs to understand the system being certified.

To address the above problem, we propose a novel technique that guides system designers in establishing a sound relationship between the domain model for a safety-critical application and the evidence model for a certification standard. Our approach makes use of UML profiles. This enables us to build upon mature MDE technologies and tailor them for our specific needs, particularly for specifying and automatically checking the constraints that must hold for compliance with safety standards.

More precisely, we begin with developing a profile based on the conceptual model of a given standard. The profile is then augmented with verifiable constraints that help system suppliers to systematically relate the concepts in the standard to the concepts in the application domain. The resulting relationship provides a clear route for the supplier to demonstrate how their development artifacts can be used for showing compliance to the standard. We illustrate our approach by showing how the concepts in the domain of sub-sea control systems can be related to the evidence requirements in the IEC61508 standard, which is one of the most commonly used certification standards for control systems.

The remainder of this paper is structured as follows: In Section II, we review background information for the paper and in Section III we outline our overall methodology for creating certification evidence for compliance. In Section IV, we present our UML profile for IEC61508 and in Section V we discuss how the profile can be used for the creation of certification evidence. Section VI compares our work with related work. Section VII concludes the paper with a summary and suggestions for future work.

II. Background

In this section, we briefly introduce safety certification and how the evidence for standards compliance can be structured through conceptual modeling, and UML profiles.

A. Safety Certification

Safety-critical systems are typically subject to a safety certification process. The aim of certification is to provide assurance that the system has been deemed safe for use in a specific environment. This is usually carried out by a third-party certification body. The certification is usually based on a specific standard applicable to the domain in which the system is operated, e.g., there is the general standard IEC61508 for the certification of electrical, electronic or programmable electronic systems that are used in safety-critical environments, the IEC61511 standard for the process industry [9], EN50129 [10] for railways, and NORSOK I-002 [11] for safety automation systems in the petroleum industry. All these standards present requirements for how the system should be created to ensure the quality of the end product and more specifically to ensure that the system is safe for operation. The justification for safe operation of a system is usually presented as a safety case [1].

A safety case is made up of three principal parts [1]: safety objectives, arguments, and evidence. Demonstrating the satisfaction of the objectives involves gathering evidence during the lifecycle of a system and constructing well-reasoned arguments that relate the evidence to the objectives. With the growing use and complexity of software in safety-critical systems, licensing and safety regulatory bodies increasingly require system suppliers to provide an explicit software safety case. A software safety case is a part of an overall system safety case, which provides assurance that the software elements of a system satisfy the safety aspects stated in the technical and software requirements specification of the system [12]. While the argumentation aspects of software safety cases have been studied before [1], little has been done to precisely specify the evidence that underlies software safety arguments [13]. As a result, suppliers of safety-critical software have been left without proper guidance on what evidence to collect during development. This has led to the suppliers having to recover the relevant evidence after the fact, which can be extremely costly or even infeasible. In addition, the quality of the overall safety case is bound by the quality of the weakest link. Hence, current practices for managing software safety evidence can severely limit the effectiveness of safety cases in general. In this paper, we provide a flexible methodology for systematically specifying safety evidence and establishing a precise link between the evidence and the requirements of relevant standards. This will in turn help with the construction of more definitive software safety cases.

B. Conceptual Models

In general, standards, irrespective of the domains they are targeted at, tend to be expressed as textual requirements. Since the requirements are expressed in natural language, they are subject to interpretation by the users of the standards. To make the interpretation explicit and develop a common understanding, we propose the development of a conceptual model that formalizes the evidence requirements of a given standard. Lewis [14] expresses the need for presenting

a safety case as an information model. He highlights the need for creating a formal structure for a safety case and the need to present the relationships that exist between atomic items of information resulting in a web of information that supports the safety case argument. In order to represent these relationships as required by a particular standard, we create a conceptual model that allows us to represent the main factors that need to be considered for certification and the relationships amongst them. The fundamental elements that we need to represent are 1) concepts, 2) attributes, 3) inter-concept relationships and 4) constraints. Additionally, as standards can be quite large, it is useful to have a means to divide the concepts into useful groupings. The UML [5] class diagram notation can be used to conveniently express the conceptual model. Concepts are represented as classes and concept attributes – as class attributes. Relationships are represented by associations. Generalization associations are used to derive more specific concepts from abstract ones. When an attribute assumes a value from a predefined set of possible values, we use enumerations. Finally, we use the package notation to make groupings of concepts and thus better manage the complexity [3].

C. UML Profiles

UML profiles [5] are a lightweight solution for extending the UML metamodel for a specific domain. They enable the expression of new concepts, notation and constraints by the introduction of context-specific stereotypes, attributes and constraints. Stereotypes are a means of extending a base metaclass of the UML metamodel. We extend different metaclasses to create stereotypes for the concepts, their attributes and their relationships respectively. Moreover, constraints can be defined in a profile by using the Object Constraint Language (OCL) [6] to ensure that certain semantics are maintained in the models to which the profile is applied. By using profiles the models that employ the profile are still consistent with the UML metamodel. The profile stereotypes along with constraints written in OCL provide us with the mechanism to guide the creation of the evidence requirements for a specific standard.

As we describe in the subsequent sections, we use this mechanism to create a profile of the IEC61508 conceptual model (Section IV) and then use it to create the evidence required for the certification of a sub-sea control system in the petroleum industry.

III. Methodology

We propose a methodology for assisting system suppliers in preparing for certification of their systems according to industry-relevant standards. Our methodology guides system developers in establishing a relationship between a domain model of a safety-critical application and the evidence model of a certification standard. We make use of UML profiles which allows us to build upon mature MDE technologies

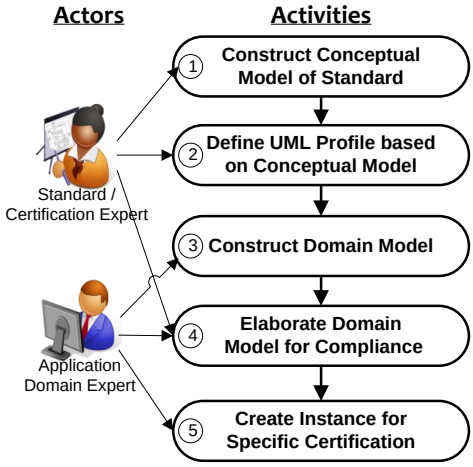

Figure 1. Methodology for the Creation of Evidence of a Safety Standard.

for specifying and automatically checking the constraints that must hold for compliance with safety standards. The methodology consists of five main steps as shown in Figure 1, which will be illustrated by our case study.

The first step is the creation of a conceptual model of the standard according to which a system needs to be certified. This process involves interpreting the text of the standard and picking the main concepts presented in it, any attributes the concepts may have, and any inter-concept relationships. Note that making these concepts and their relationships explicit is an important aspect of compliance [13], [15], [14].

The second step is the creation of a UML profile based on the conceptual model. The profile is in turn used for stereotyping the elements of a domain model created in step three. Broadly, a domain model is a representation of the core concepts in an area of interest. In this paper, we use the term domain model to refer to concepts that represent the physical and abstract components of a *family* (class) of systems in a particular application area (e.g., sub-sea control systems), the environment in which this family of systems function, and the key artifacts built throughout development. An example of product family [16] is a Fire and Gas Protection system that will consist of sensors being used to detect fire or combustible gas, a controller that does processing based upon the input from the sensors and then deploys certain actuators such as sprinklers or dampers. This is a generic description of a class of systems – each variant of the system will have very specific types of sensors and actuators with specific actions that should take place upon the detection of fire or gas. Following the norm in MDE, we assume domain models are represented as UML class diagrams[17]. Using a profile makes it possible to establish a concrete link between the evidence requirements of a given standard and a domain model. In this paper, we do not concern ourselves with the construction of domain models. Good references and guidelines already exist [17].

When a stereotype from the profile of a given standard is applied to a domain model element, it shows how that

element fulfills the requirements from the standard. The profile is created by mapping the concepts in the conceptual model as extensions of the metaclass 'Class' in the UML metamodel, the attributes of the concepts are made into attributes of the class to which the stereotypes are applied to, the relationships between the concepts are mapped as extensions of the metaclass 'Association'. Enumerations are used for describing either standard-specific or user-specific data types. OCL constraints are added to the stereotypes to ensure certain properties of the stereotypes as well as to guide system developers in elaborating the domain model for the system being developed.

The fourth step requires the elaboration of the domain model. Specifically, elaboration means the application of the profile stereotypes to the appropriate domain model elements, and refining the domain model so that it satisfies the OCL constraints of the stereotypes. These refinements could include the addition of new domain model elements or making changes to the existing ones (e.g., adding new attributes, revising multiplicities).

The process starts by applying a stereotype to the domain model itself, stating which standard the domain model needs to comply with. If the standard of interest is IEC61508, the stereotype could be `IEC61508Model`. The OCL constraints associated with this stereotype will start the guidance process for augmenting the domain model with other stereotypes. This in turn may require the domain model to be updated so that the stereotype constraints are satisfied. Each new stereotype applied will have further constraints that will need to be satisfied for the model to be valid. This chain of constraints will guide the elaboration of the domain model, so that it will cover all aspects that are necessary for certification. Ultimately, that elaborated domain model will represent a precise specification for the safety evidence and explicit links to the standard's requirements.

Finally, for certifying a specific system (variant) from a product family, step five in Figure 1 is performed. This step creates an instantiation of the UML class diagram representing the elaborated domain model. In other words, an object diagram of the domain model is built to represent the specific properties of a system variant.

Steps one and two and four need input from an expert who understands the certification process and the standard to which the system will be certified; whereas, in steps three, four and five the knowledge of an application domain expert is required. Finally, we note that steps one and two are carried out once per standard; steps three and four are done once per product family; and step five is performed once for each variant that is subject to certification.

We exemplify this whole process by creating a conceptual model and then profile of the IEC61508 standard. We present the profile in Section IV and show how the domain model of a sub-sea control system is elaborated with the application of stereotypes from the IEC61508 profile in Section V.

IV. The IEC61508 Profile

A. IEC61508 Standard

The IEC61508 standard presents requirements to facilitate the development of safety-related electrical/electronic/programmable electronic systems (E/E/PES). The goal of the standard is to ensure the functional safety of safety-related E/E/PES systems. Functional safety is a component of overall safety, for example, the activation of an alarm in response to a fire detection by a control system is a functional safety measure, whereas the use of fire resistant walls to control the spread of fire is not, whilst it is still a part of overall safety measures. A function that a control system performs to ensure that the system remains in a safe state is referred to as a safety function. Each safety function specifies which safety objective is to be achieved (safety function requirement) and the level of integrity with which the safety function is implemented (safety integrity level).

To achieve the required level of safety, the standard recommends the use of a safety lifecycle. The lifecycle should contain certain activities such as a hazard analysis and risk assessment to determine the hazards that can occur and the risks that they pose. Together, these activities determine what has to be done to avoid hazardous situations (derivation of safety requirements) and the level to which safety has to be provided (derivation of safety integrity levels). The derived safety requirement are allocated to either certain functions of a designated E/E/PE safety-related system, other technology safety-related systems, or to external risk reduction facilities. The IEC61508 standard is only concerned with the allocations made to the E/E/PE system. Once the requirements have been allocated, the realization of the system begins for both the hardware and software aspects of the E/E/PE system. The activities concerned with the installation and commissioning, operation and maintenance, and the final overall safety validation of the system begin alongside the realization of the system.

B. IEC61508 Conceptual Model

The conceptual model for the IEC61508 standard was built in our previous work on IEC61508 [3]. As mentioned in Section II, we use UML class diagrams to create the conceptual model where concepts are represented as classes and concept attributes as class attributes; relationships are represented by associations. When an attribute assumes a value from a predefined set of possible values, we use enumerations. Finally, we use the package notation to make groupings of concepts and thus better manage the complexity.

The conceptual model has a total of 10 packages, containing abstractions for modelling the main concepts of IEC61508. We briefly explain each package. For more details, see [3]. The `System Concepts` package describes

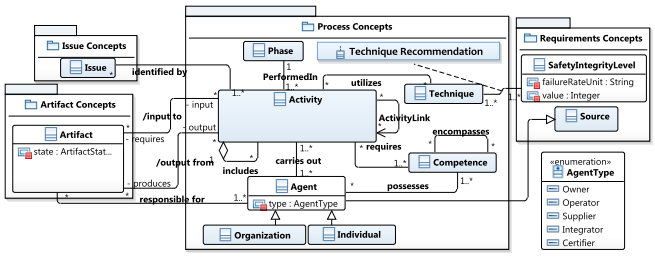

Figure 2. Process Concepts Package of the IEC61508 Conceptual Model

the breakdown of the system and reflects both hardware and software concepts; the `Hazard Concepts` package captures the abstraction for describing the hazards and risks for the system and leads to the specification of safety requirements; the `Requirements Concepts` package captures the requirements for creating, operating, maintaining and decommissioning control systems; the `Process Concepts` package is for describing the development process for creating the system; the `Artifact Concepts` package is for describing the different types of artifacts created as supporting evidence during the development of the system; the `Guidance` package is for describing the other standards and recommended practices that will be used to develop the system, the `Issue Concepts` package is for describing the defects or enhancements that may give rise to changes; the `Configuration Management Concepts` package is for describing the unique versions for all the components that make up the system, the `Justification Concepts` package to capture the assumptions and rationale behind the various decisions that are made during development; and the `Domain-Specific Concepts` package for capturing the enumerations for concept attributes in other packages (e.g., requirement type, artifact state).

As a small example, we show in Figure 2 part of the conceptual model for specifying the process of development. The main concept in this package is the concept of activity, this is a unit of work which has specific artifacts as defined inputs and outputs. An activity can be decomposed into further activities and is performed within a larger unit of work called a phase. A phase defines a means to manage a related set of activities together and a number of phases are used to manage the development of the entire system. An activity may have one of more agents that perform it and there are certain techniques that are utilized to carry it out. The techniques selected for an activity are based on the safety integrity level that needs to be achieved. Thus, module testing alone may be sufficient for a low level of safety integrity but at higher levels, testing along with formal proofs may be required. The agents that carry out the activity need to possess the competence that the activity requires and may be either individuals or organizations. The concepts related to the process are together used to show that competent agents have created the system in an organized manner. Details for the other packages can be found in [3].

C. IEC61508 Profile

The IEC61508 profile is a means of showing how a system fulfills the requirements of the IEC61508 standard. The stereotypes are based on the conceptual model of the standard that was created in our earlier work [3] and highlight how a particular aspect of the system fulfils the IEC61508 standard.

The IEC61508 profile consists of the following:

- 1 stereotype that extends the metaclass `Model`, that characterizes the base domain model as being certified to the IEC61508 Standard..
- 4 stereotypes that extend the metaclass `Package`, that is used to organize the evidence at a high level.
- 54 stereotypes that extend the metaclass `Class`, that are used to characterize the evidence elements.
- 53 stereotypes that extend the metaclass `Association`, that are used to characterize the relationships between the evidence elements.
- 6 stereotypes extend the metaclass `Property`, that are used on role names of certain associations.

All stereotypes have documentation attached to them that explains what the stereotype is meant for and OCL constraints that perform two main functions: they ensure that the stereotypes are used in the way intended, and provide guidance to the user as to which stereotypes need to be used in the first place, e.g., the stereotype `Activity` has constraints that ensure that elements with the stereotype `Agent` also exist in order to show who is performing the activity. In this way we create the web of evidence information mentioned in Section II. We use OCL constraints for a number of purposes:

1) To ensure that mandatory aspects of the standard are accounted for.
2) To ensure the correct type of stereotypes at the two ends of associations.
3) To ensure that elements with certain stereotypes are connected to other specific elements.
4) To ensure that elements with certain stereotype have specific properties - this helps when creating instances of the model.
5) To help with the creation of user-defined enumerations defined in the conceptual model.

As it would be difficult to show all stereotype of the profile within the size constraints of this paper, in Figure 3 we show a fragment of the profile corresponding to the process concepts package discussed earlier along with the stereotypes applied at the model and package level. We also use this fragment to show examples of the five types of constraints mentioned above.

As we stated earlier, the IEC61508 profile is meant to be applied to a domain model of the system to be certified. The first stereotype to be applied is at the model level: the `IEC61508Model` stereotype, which

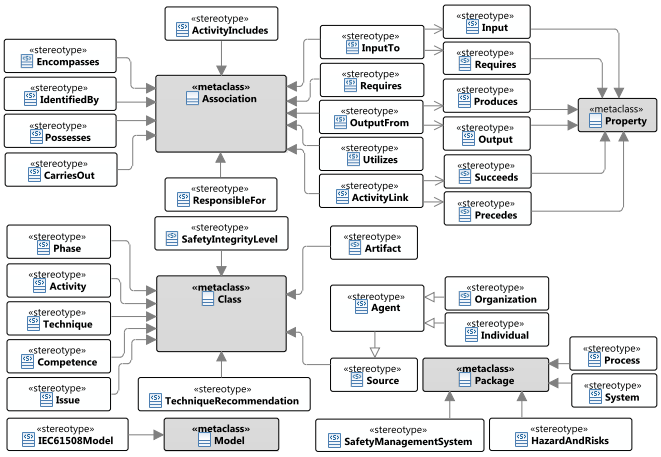

Figure 3. IEC61508 Profile Fragment for the System Development Process

starts the incremental guidance process about which types of evidence to create. This stereotype has attached to it the OCL constraints that ensure that at a minimum four specific packages exist in the model with the stereotypes `Process`, `System`, `SafetyManagementSystem` and `HazardsAndRisks` (constraints of type 1). As an example, we show the constraint on this stereotype for ensuring that the `HazardsAndRisks` stereotype exists on a UML package in the model (for the sake of brevity we have omitted the name and context of the constraints shown):

```
self.base_Model.allOwnedElements()-> exists(e |
e.oclIsTypeOf(uml::Package) and not e.getAppliedStereotype
('IEC61508Profile::HazardsAndRisks').oclIsUndefined())
```

The keyword `self` refers to the element being constrained, in this case the `IEC61508Model` stereotype. Properties and attributes of an element are referenced using the dot notation. The `base_Model` reference is used to access the model to which the stereotype has been applied, in this case the domain model. The `allOwnedElements` is an operation that returns all the elements in the model. `Exists` is an OCL operation that will check that least one element in a collection of elements satisfies the given constraint. The constraint in the `exists` clause specifies that at least one element is of type `Package` using the operation `oclIsTypeOf` and that this element also has the stereotype `HazardsAndRisks` applied to it (using the operation `getAppliedStereotype`).

The rationale behind requiring these packages comes from the IEC61508 standard. The IEC61508 standard advocates a risk-based approach for determining the required level of safety measures for safety-relevant systems. Hence the need for the `HazardsAndRisks` package. Risks can only be determined based upon the hazards that will exist when the system is used, thus it is important to have a breakdown of the system, bearing in mind both the hardware and software aspects of the system as well as the role of human users. This breakdown will be kept in the `System` package. The standard also put emphasis on having clearly specified technical and management activities and a clear identification of all responsible persons within the organization that perform these activities. The management information is kept in the `SafetyManagementSystem` package whereas the technical activities are specified within a safety life-cycle and kept in the `Process` package.

The standard does not require a specific kind of life-cycle but does state which activities should be carried out and which artifacts should be produced. Thus, we have the stereotype `Phase` to model the life-cycle and the stereotype `Activity` to model the activities. The `Process` package has a constraint (type 1) on it that specifies that it should contain in it elements with the stereotype of `Phase`:

```
self.base_Package.allOwnedElements()-> exists(el:Element |
el.oclIsKindOf(uml::Class)and not(el.getAppliedStereotype(
'IEC61508Profile::Phase').oclIsUndefined()))
```

The `Phase` stereotype has a constraint attached to it that states that every phase must have at least one `Activity` defined for it (a type 3 constraint). This means that there must be elements that have the stereotype `Activity` in the same package and be attached to the element with the stereotype `Phase`. This is done through two different constraints, one on the class stereotype `Phase` and the other on the association stereotype `PerformedIn` (see Figure 2). On the stereotype `Phase`, we have a constraint that states that there should be an association with a stereotype `PerformedIn` originating from the element that has this stereotype:

```
self.base_Class.ownedAttribute-> collect(c:Property |
c.association)-> select(a:Association |
not a.getAppliedStereotype('IEC61508Profile::PerformedIn')
.oclIsUndefined())->size()>0
```

On the stereotype `PerformedIn`, there is the constraint that states that this stereotype can only be applied to an association that is between a pair of elements that have the stereotypes `Activity` and `Phase`, respectively (a type 2 constraint):

```
self.base_Association.memberEnd-> select(p:Property not
(p.class.getAppliedStereotype('IEC61508Profile::Activity')
.oclIsUndefined()))->size()=1
and
self.base_Association.memberEnd-> select(p:Property| not
(p.class.getAppliedStereotype('IEC61508Profile::Phase')
.oclIsUndefined()))->size()=1
```

An activity can include sub activities or it can be linked to another activity by either preceding or succeeding it, all these relationships are modelled by the stereotypes `ActivityIncludes` and `ActivityLink`, along with its properties `Precedes` and `Succeeds`. Activities are to be performed by competent agents using recommended techniques, that are based upon the safety integrity level allocated to a component. All these aspects are modelled using the stereotypes `Agent`, `Requires`, `Competence` along with `CarriesOut`, `Possesses`, `Technique` and `TechniqueRecommendation`. An activity may require certain artifacts as input and upon completion produce certain artifacts as outputs. The stereotypes `Artifact`, `InputTo`, `OutputFrom`, `Requires`, `Produces`,

`Input` and `Output` are used to model these concepts. Constraints on the stereotype `Activity` ensure that for every activity, the agent that carried out the activity is defined as well as the output from the activity. Constraints are also used to create properties for the elements on which stereotypes have been applied or for creating user-defined types, e.g., An element with the `Artifact` stereotype applied to it should have a property called 'State' of type 'ArtifactStateType' which is a user defined enumeration (this constraint combines both type 4 and 5 constraints):

```
self.base_Class.ownedAttribute-> one(p:Property|
p.name='State' and p.type.name='ArtifactStateType' and
p.type.oclIsTypeOf(uml::Enumeration))
```

These types of constraints allow the user to define domain-specific values for the enumeration, the profile only gives the name and type of the property. An advantage of using OCL constraints is that they can be automatically checked using any OCL validation engine, thus providing a means of efficiently checking large amounts of evidence in terms of completeness and consistency. The stereotypes together with the constraints defined on the stereotype guide the user in creating an information model of the evidence necessary for certification. This in turn will enable the systematic collection and automated analysis of evidence. In the next section, we show how the IEC61508 profile can be used to manage the certification evidence for a sub-sea control system in the petroleum industry.

V. Case Study: Application of the IEC61508 Profile to the Sub-sea Control Domain

To validate our approach, we have applied it for guiding the construction of a domain model for sub-sea control systems in compliance with IEC61508, and to partially create the evidence for a specific variant of the system. Our sub-sea domain concepts were defined in close consultation with experts in a large maritime and energy company and based on a reading of the relevant literature where the architecture and the components of sub-sea systems (including the control software operating on them) are described [7], [8], [11], [18]. Due to space constraints, we are unable to show the entire domain model or to go through all the guidance steps provided by the profile. Instead, we will focus in this section on a fragment of the domain model, shown in Figure 4, and illustrate how our proposed approach is applied in a concrete way over this fragment.

In a sub-sea system, the wellhead attaches to the sub-sea oil or gas well and interfaces to the drilling and other production equipment housed in what is known as a Christmas Tree (CT). A CT in this context is an assembly of control valves, pressure gauges, and chokes put on the top of a well to control the flow of oil and gas once the well drilling operation has been completed. The CT controls the flow of oil or gas to the manifold which provides the connections to direct the oil or gas away from the production system.

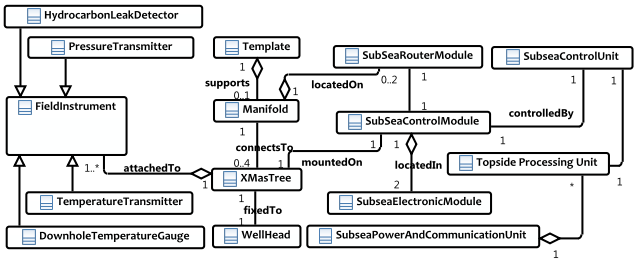

Figure 4. A Domain Model Fragment of a Sub-Sea Control System

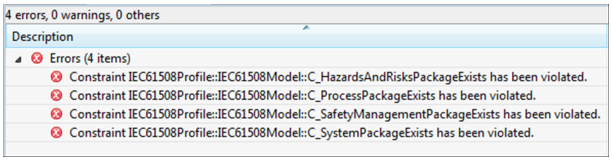

Figure 5. Error Report Showing the Violated OCL Constraints of the IEC61508Model Stereotype

All this equipment is anchored to the seabed via a structural frame called the template. Mounted on the CT is the Sub-sea Control Module (SCM) that receives commands from the Sub-sea Control Unit (SCU) that is executing in the Topside Processing Unit (TPU) located in the Sub-sea Power and Communication Unit (SPCU). The SCM also sends signals from the sub-sea instruments to the SCU. The signals are sent from the SCM via the Sub-sea Router Module (SRM) to a router in the SPCU that passes the signal to the TPU. A more complete description of these components can be found in [7], [8].

Once the initial domain model has been created, the `IEC61508Model` stereotype is applied to the domain model and the OCL constraints of this stereotype are validated. Figure 5 shows the beginning of the guidance process for creating the evidence. The first thing required is the creation of four packages that have the stereotypes: `Process`, `System`, `SafetyManagementSystem` and `HazardsAndRisks`. The packages themselves can be named using the supplier's own terminology, but the specified stereotypes need to be applied. Once these stereotypes have been applied, the next set of (failed) constraints will provide guidance on what stereotypes to apply next.

Figure 6 shows that five constraints have failed once the package stereotypes have been applied. Hazards have not yet been identified in the `HazardsAndRisks` package; phases have not yet been identified in the `Process` package; agents and their competence have not been identified in the `SafetyManagementSystem` package, and blocks have not been identified in the `System` package. As an example, we will show the application of the stereotypes that identify the system components.

The SCU controls and monitors the sub-sea wells through the operator station, so the stereotype `SoftwareBlock` is applied to this element. The system being controlled is the SCM that contains the Sub-sea Electronic Module (SEM) that links to the different instruments. Thus, the stereotype

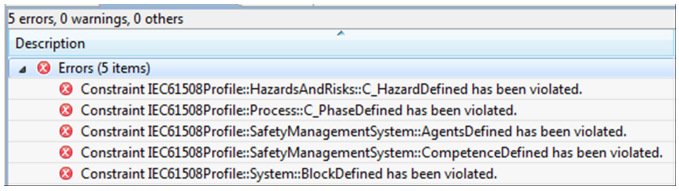

Figure 6. Error Report Showing the Violated OCL Constraints After Application of the Package Stereotypes

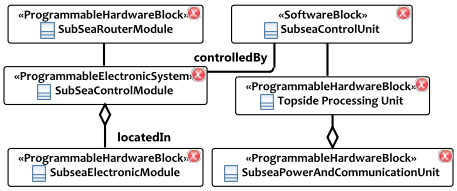

Figure 7. Fragment of the Domain Model After Application of the System Stereotypes

`ProgrammableElectronicSystem` is applied to the SCU and the stereotype `ProgrammableHardwareBlock` to the rest of the elements. If we now validate the model then, we get new constraints that are violated as the model is now missing further information.

In Figure 7, all the elements have a cross in their upper right-hand corner. These element have failed the OCL constraint validation. For brevity, we show in Figure 8, the errors generated for the SCU only; similar errors are also generated for the other elements.

All system blocks need to have requirements allocated to them. In this model, the allocated requirements have not been added to the model, leading to the violation of constraint regarding requirements. Blocks also need to have unique identifiers, which have not been yet added to the elements – in the petroleum industry, every components of a system has a unique identifier called a tag. The standard also recommends version control of the system components - a version attribute has not yet been added to the elements either, thus the violation of that constraint. If we add the elements to satisfy these constraint violations, then we get the model depicted in Figure 9. Two requirements at the system level have been added to the model - SReq1 and SReq2. They model two different kinds of requirements that are common on sub-sea control systems: the shutdown of parts of the system due to an emergency and the monitoring of the status of certain instruments. All requirements are kept in an artifact called the SystemSoftwareRequirements.

A user-defined enumerated type has also been added. A constraint on the `SoftwareBlock` stereotype requires the need to show the decomposition level of the software. The constraint specifies the name of the attribute ('Level') and

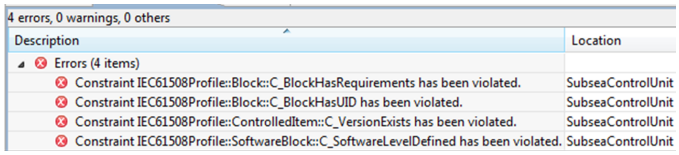

Figure 8. Error Report Showing the Violated OCL Constraints for the Subsea Control Unit

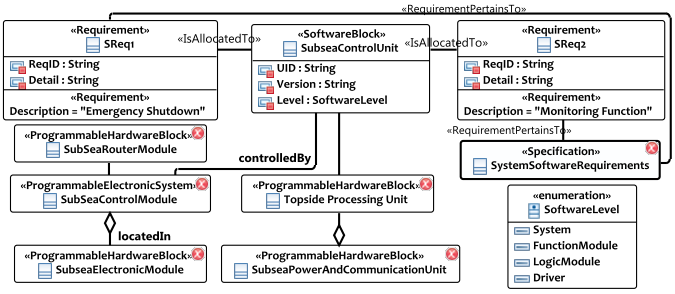

Figure 9. Fragment of the Domain Model After Application of the System Stereotypes

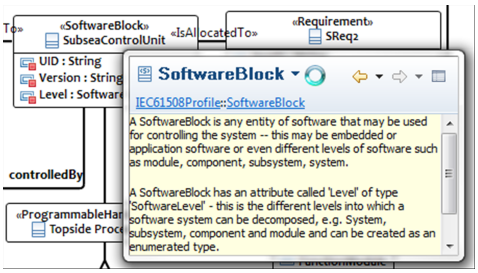

Figure 10. Documentation for a Stereotype

the type ('Enumeration'). The actual literal values are set by the user as relevant to their industry. In this case the literals used are 'System', 'FunctionModule' , 'LogicModule' and 'Driver' - this is the breakdown that is most commonly used in the sub-sea industry and hence the user is able to use appropriate domain terminology. Attached to each stereotype is documentation that can help the user to understand how to use a particular stereotype, as shown in Figure 10. Thus all stereotypes are documented in this way to aid the user in elaborating the domain model with the necessary stereotypes. The documentation is useful is giving general guidance as the constraints alone would not be sufficient. The creation of the profile and the models and the validation of constraints can be performed using a UML modelling tool, e.g., Rational Software Architect [19]. We used this tool in our case study here.

As more stereotypes are added and the constraints are evaluated, the web of evidence is created for a particular product family. Once the elaboration of the domain model is complete with all the generic information, the instance for a particular system variant can be created. In Figure 11, we show a small instance model conforming to the domain model that we present in Figure 9. In this particular instance there is one template and one manifold. The manifold has two CT structures on it, each connected to a wellhead. There are two SCMs, both controlled by a single SCU. The unique Id given to the SCU is 'T1823a', the version of the software for the SCU is 1.2 and this is the system level software version. For hardware equipment, the version would store the model number or serial number of the piece of equipment. Three requirements are shown: RQ1.121 and RQ1.212 are instances of SReq1 which is the requirement in the domain model concerning emergency shutdown of the system. There

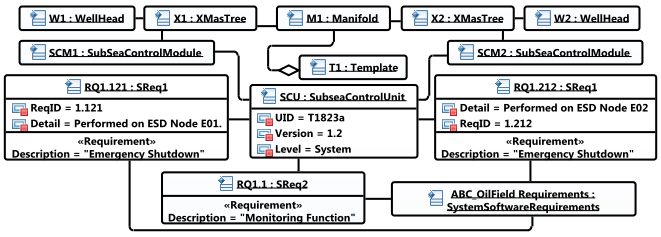

Figure 11. Instance Model Created from the Elaborated Domain Model

are two instances in the actual system of this requirement to deal with the shutdown of the two wells independently. A further requirement, RQ1.1 is shown that would be for monitoring the status of some piece of equipment - there would be multiple instances of this requirement in the system to monitor the various instruments. All requirements are kept in the specific requirements artifact called theABC_OilField Requirements. The UID, Version, Level, ReqID and Detail attributes were added to the elements due to constraints on the stereotypes, this allows specific values to be set for these elements in the instance model. The stereotypes can have attributes as well, as in the case of 'Description' for the `Requirement` stereotype. The value for this stereotype is set in the domain model and does not change in the instance model. The use of an elaborated domain model for the certification evidence of a product family and the instance model for the evidence for a particular variant allows a high degree of reuse and helps to reduce the effort needed in creating the evidence for each particular variant.

VI. Related Work

There are two areas of related work that are important to mention regarding our work: the creation of electronic safety cases and the use of UML for the development of safety-critical systems.

The need for constructing electronic safety cases has been identified by [20] and [14] in order to manage the complexity and large amounts of information that needs to be kept for a safety case. Lewis [14] calls for an underlying information model to manage the complex links that exist between the various pieces of safety evidence. We propose an automated methodology to do so, in the context of the IEC61508 standard. Our conceptual model provides the underlying information model for the standard and our profile provides a practical mechanism for using this information in order to create the relevant artifacts for a safety case. Cockram and Lockwood [20] present a hypertext linked approach to linking all the pieces of information for a safety case in the form of a proprietary tool. We on the other hand use model-driven engineering technologies, and in particular UML profiles, to aid with the creation of the safety case for a particular standard. The profile can be exported and used in any UML modelling tool. This allows one to keep a set of inter-related information items which can be viewed directly from the tool, or can be transformed into different views by the use of reporting tools.

The use of model-based technologies is gaining pace in industry. Especially, UML is increasingly used in the development of safety-critical software. The Object Management Group (OMG) have standardized the UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [21] and the UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms (QFTP) [22]. Both these profiles are used for modelling the real-time and performance properties of safety-critical systems. Similary, Berkenkotter [23] and Hannemann [24] have created a profile for the railway domain that aids the design and verification of interlocking functionality. However, neither of these are meant to characterise the evidence requirements of a standard to which safety-critical systems are certified.

A profile that deals with certain aspect of certification is proposed by Zoughbi et. al. [25]. Their profile enables the direct addition of certification information to software models, for compliance with the RTCA DO-178B standard [26] used in commercial and military aerospace software. However, this profile is targeted at maintaining traceability between requirements, design and code, which is a part of the requirements of the RTCA DO178B standard. The profile that we propose deals with a different standard, IEC61508, and takes into account not only evidence regarding requirements and design but also with the wide range of concepts related to the management of the development process in safety-critical systems.

Huhn and Hungar [27] discuss the proliferation of UML in the model-based development of safety-critical software and mention the profiles discussed above. They propose a development process where models form an integral part of the development of a safety-critical system. However, they do concede that the use of models for the certification aspect has not been adequately addressed. Our profile is a starting point for addressing this gap.

Recently, the OMG has put forward a new proposal, called the Software Assurance Evidence Metamodel (SAEM) [28], for managing safety assurance evidence. The SAEM is a standard-independent metamodel and directed towards linking the certification evidence to safety claims and the evaluation of these claims subject to the evidence. The methodology that we propose uses a UML profile for characterizing the evidence of a specific standard. To perform the same task, the SAEM model will still require a definition of the specific evidence needed by a particular standard (perhaps based on a conceptual model was we have proposed). On the other hand, a profile of the SAEM could be incorporated into our methodology and cover both the evidence requirements for compliance to IEC61508, as well as the evaluation of the evidence to ensure that it is sufficient to substantiate the safety claims. Together this could be a means to further the

field of model-based certification.

Finally, we have used UML profiles of safety related standards in prior work [4], where we ensure that a generic standard can be specialized for a particular domain in a systematic manner. In contrast to this current paper, profiles were used in [4] as a way to keep track of the relationships between a generic standard and a sector-specific one.

VII. Conclusion and Future Work

In this paper we showed how to use model-driven engineering principles and technology to specify and analyze safety evidence in order to show conformance to a safety standard. We start by establishing a sound relationship between a domain model of a safety-critical application and the evidence model of a certification standard. We do this by capturing the relevant standard as a conceptual model using a UML class diagram and using this as a basis for creating a UML profile. The profile is augmented with constraints to aid system suppliers in systematically relating the concepts in the standard to the concepts in the application domain. The profile is then applied to a domain model of a safety-critical application aiding system suppliers in clearly demonstrating how the development artifacts of their system fulfill the compliance requirements of a standard. Constraints can be automatically checked to ensure full compliance. We illustrate our approach by presenting an excerpt of a case study that we are conducting to show how the concepts in the domain of sub-sea control systems can be related to the evidence requirements in the IEC61508 standard.

In future work we plan to complete the case study and assess the cost-effectiveness of our methodology in the context of certification. This would mean, creating a full instantiation of a system with all the certification information included. This would allow us to compare our methodology with the current industry practice of preparing for certification. We intend to check how complete our set of certification information is - how many details do we include when using our methodology that are missed with current practice and how efficient is our methodology in collating all the required information compared with current practice. We also plan to extend our methodology so that we can express how a repository of evidence information can address the certification of a system to multiple inter-related standards. Finally, we would like to extend our work to include the evaluation of evidence as proposed by the SAEM.

References

[1] T. P. Kelly, "Arguing safety – a systematic approach to managing safety cases," 1998.

[2] International Electrotechnical Commission, "Functional safety of electrical / electronic / programmable electronic safety-related systems (IEC 61508)," 2005.

[3] R. K. Panesar-Walawege, M. Sabetzadeh, L. Briand, and T. Coq, "Characterizing the chain of evidence for software safety cases: A conceptual model based on the IEC 61508 standard," in *ICST*, 2010, to appear.

[4] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand, "Using uml proles for sector-specic tailoring of safety evidence information," Simula Research Laboratory, http://simula.no/publications/Simula.simula.600, Tech. Rep. 2011-10, 2011.

[5] "UML 2.0 Superstructure Specification," August 2005.

[6] "OMG Object Constraint Language."

[7] ISO, "Petroleum and natural gas industries - design and operation of subsea production systems (ISO 13628)," 2005.

[8] Y. Bai and Q. Bai, *Subsea Engineering Handbook*. Elsevier, 2010.

[9] IEC, "Functional safety - safety instrumented systems for the process industry sector (IEC 61511)." 2003.

[10] EN, "Railway Applications Safety-related electronic railway control and protection systems." 1999.

[11] Norwegian Technology Centre, "Safety and automation system (SAS)," 2001.

[12] UK Ministry of Defence, "Defence standard 00-55, requirements of safety related software in defence equipment (DS 00-55)." 1997.

[13] D. Jackson, M. Thomas, and L. Millett, *Software for Dependable Systems: Sufficient Evidence?* National Academy Press, 2007.

[14] R. Lewis, "Safety case development as an information modelling problem," in *Safety-Critical Systems: Problems, Process and Practice*. Springer, 2009, pp. 183–193.

[15] P. Bishop and R. Bloomfield, "A methodology for safety case development," in *Safety-Critical Systems Symposium*. Springer, 1998.

[16] K. Pohl, G. Böckle, and F. van der Linden, *Software product line engineering - foundations, principles, and techniques*. Springer, 2005.

[17] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall, Oct. 2004.

[18] Norwegian Oil Industry Association, "Application of IEC61508 and IEC61511 in the Norwegian Petroleum Industry," 2004.

[19] "IBM Rational Software Architect," http://www.ibm.com/developerworks/rational/products/rsa/.

[20] T. Cockram and B. Lockwood, "Electronic safety cases: Challenges and opportunities."

[21] OMG, "UML profile for modeling and analysis of real-time and embedded systems (marte)," http://www.omg.org/spec/MARTE/1.0/, 2009.

[22] ——, "UML profile for modeling quality of service and fault tolerance characteristics and mechanisms specification," http://www.omg.org/spec/QFTP/1.1/, 2008.

[23] K. Berkenkötter, "Ocl-based validation of a railway domain profile," in *MoDELS Workshops*, 2006, pp. 159–168.

[24] K. Berkenkötter and U. Hannemann, "Modeling the railway control domain rigorously with a uml 2.0 profile," in *SAFECOMP*, 2006, pp. 398–411.

[25] G. Zoughbi, L. Briand, and Y. Labiche, "Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and uml profile," *Software and Systems Modeling*, pp. 1–31, 2010.

[26] "DO-178B: Software considerations in airborne systems and equipment certification," Radio Technical Commission for Aeronautics (RTCA), 1982.

[27] M. Huhn and H. Hungar, "8 uml for software safety and certification," in *Model-Based Engineering of Embedded Real-Time Systems*, ser. Lecture Notes in Computer Science, H. Giese, G. Karsai, E. Lee, B. Rumpe, and B. Schtz, Eds. Springer Berlin / Heidelberg, 2011, vol. 6100, pp. 201–237.

[28] OMG, "Software Assurance Evidence Metamodel (SAEM)," http://www.omg.org/spec/SAEM/, 2010.