Supporting the Verification of Compliance to Safety Standards via Model-Driven Engineering: Approach, Tool-Support and Empirical Validation

Rajwinder Kaur Panesar-Walawege\textsuperscript{a,b,*}, Mehrdad Sabetzadeh\textsuperscript{c}, Lionel Briand\textsuperscript{c,b}

\textsuperscript{a}University of Oslo, Norway  
\textsuperscript{b}Simula Research Laboratory, Norway  
\textsuperscript{c}SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

Abstract

Context. Many safety-critical systems are subject to safety certification as a way to provide assurance that these systems cannot unduly harm people, property or the environment. Creating the requisite evidence for certification can be a challenging task due to the sheer size of the textual standards based on which certification is performed and the amenability of these standards to subjective interpretation.

Objective. This paper proposes a novel approach to aid suppliers in creating the evidence necessary for certification according to standards. The approach is based on Model-Driven Engineering (MDE) and addresses the challenges of using certification standards while providing assistance with compliance.

Method. Given a safety standard, a conceptual model is built that provides a succinct and explicit interpretation of the standard. This model is then used to create a UML profile that helps system suppliers in relating the concepts of the safety standard to those of the application domain, in turn enabling the suppliers to demonstrate how their system development artifacts comply with the standard.

Results. We provide a generalizable and tool-supported solution to support the verification of compliance to safety standards. Empirical validation of the work is presented via an industrial case study that shows how the concepts of a sub-sea production control system can be aligned with the evidence requirements of the IEC61508 standard. A subsequent survey examines the perceptions of practitioners about the solution.

Conclusion. The case study indicates that the supplier company where the study was performed found the approach useful in helping them prepare for certification of their software. The survey indicates that practitioners found our approach easy to understand and that they would be willing to adopt it in practice. Since the IEC61508 standard applies to multiple domains, these results suggest wider applicability and usefulness of our work.

*Corresponding author

Email addresses: rpanesar@simula.no (Rajwinder Kaur Panesar-Walawege), mehrdad.sabetzadeh@uni.lu (Mehrdad Sabetzadeh), lionel.briand@uni.lu (Lionel Briand)

Preprint submitted to Information and Software Technology January 29, 2013
1. Introduction

Safety-critical systems are often subject to a stringent safety certification process, aimed at providing an assurance that a system is deemed safe by a certification body. In order to structure this assurance process and the related assurance artifacts, there are certification standards that set out the requirements that system suppliers must meet. These standards can be generic standards that apply to generic technologies used across many domains or they can be very specific to a particular technology in a specific domain. There is generally overall agreement within an industry as to which standards are applicable. The suppliers of safety-critical systems are then required to present the necessary evidence to show compliance to the relevant safety standards.

Showing compliance to safety standards proves to be a very challenging task due to the fact that these standards are presented as very large textual documents that are amenable to subjective interpretation. Without an explicit interpretation of the standards, the system suppliers run the risk of missing crucial details that should have been recorded during system development. This would result in them having to reconstruct the missing evidence at the time of certification – an often expensive and time-consuming endeavour. On the certifier’s side, poorly-structured and incomplete evidence often leads to significant delays and loss of productivity, and furthermore, does not allow the certifier to develop enough trust in the system undergoing certification. It is therefore very important to devise a systematic approach, which is supported by effective automation, to specify, manage, and analyze the safety evidence necessary to demonstrate compliance to standards.

In this article, we present an approach for assisting system designers in relating the concepts of their application domain to the evidence requirements of the standards that apply to their domain. This work is motivated by a real and observed need during our work with safety-critical system suppliers. The majority of the evidence artifacts that suppliers create for certification are based on the concepts of the application domain, as opposed to the concepts of the certification standards. There is a need to link the concepts used in the application domain to those used in the standard. The absence of such a link 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 application domain concepts is covering all the evidence aspects mandated by the standard.

To provide 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 types of equipment. A sub-sea production 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 and closing valves that control the oil well. These commands are sent from the Master Control
Station (MCS) which is software that runs on the oil rig in what is called the Topside Processing Unit (TPU) [15, 1]. 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 MCS. The correlation of these simple pieces of information provides clarification to the certifier who needs to understand the system being certified.

**Solution Overview.** To address the above problem, we propose a solution based on MDE for systematically guiding system designers in establishing a sound relationship between a domain model of a safety-critical application and the evidence model for a certification standard. Our proposed approach makes use of mature MDE technologies which we tailor for our specific needs. We use UML [23] class diagrams for creating domain models of the application and conceptual models of the safety standard. We then use the extension mechanisms of UML and create a UML profile of the safety standard based on its conceptual model. The profile is then augmented with verifiable constraints, written in the Object Constraint Language (OCL) [24], that help system suppliers in systematically relating the concepts in the standard to the concepts in the application domain. In this manner, we use existing MDE technologies and tools, that are generally used for system development, and tailor them for aiding system suppliers in demonstrating how their system development artifacts comply with the requisite safety standards. In our work, we have used Rational Software Architect (RSA) [11] to provide tool support. Other modelling tools with the minimum criteria that we define in the paper in Section 4 can also be used.

**Contributions.** In order to empirically assess our approach, we have carried out a case study that shows the feasibility of the proposed approach and tooling in the energy and maritime domain. The results of the case study show that general modelling tools can be used in realizing our approach and that our approach provides adequate guidance to suppliers in creating the relevant evidence for certification. The case study is followed by a survey of domain experts to gauge their perceptions regarding our approach. The results of the survey show that the experts perceive benefit in adopting our approach in their certification work. Our approach uses general MDE techniques in a novel way and can be adapted to other standards. In summary, our contributions in this paper are: (1) A general approach that uses MDE techniques to aid preparation for certification; (2) the adaptation of general modelling tools used in system development to manage evidence for certification; (3) the application of the approach in the context of sub-sea production control systems; and (4) a survey of industry practitioners, presenting their perceptions of our approach.

Previously, we have studied different facets of the problem of safety evidence specification and management. Our prior work includes a conceptual framework for the specification of safety evidence using UML [28] and a technique for tailoring generic evidence requirements according to sector-specific needs (e.g., in the railways, avionics, and maritime and energy sectors) [30]. Further, the basic formulation of the approach presented in this current article has been previously published in a research paper at the 22th IEEE International Symposium on Software Reliability Engineering (ISSRE’11) [29]. This article brings together ideas described in these earlier papers and presents a definitive treatment of our approach for creating the necessary evidence to demonstrate standards’ compliance. Specifically, we provide a more comprehensive
description of our approach (Section 3) and tool support (Section 4), along with substantial new empirical results to show the feasibility and usefulness of our approach (Section 5).

Structure. The remainder of this paper is structured as follows: In Section 2, we discuss the challenges faced in creating certification evidence based on industry standards and the motivation for our work. In Section 3, we outline our overall approach for creating certification evidence for compliance. In Section 4, we discuss tool support, and in Section 5, we present an empirical evaluation of our approach via a case study and survey. Section 6 compares our work with related work and Section 7 concludes the paper with a summary and suggestions for future work.

2. Background and Motivation

In this section, we briefly introduce safety certification and outline the motivations for our work.

A safety-critical system is one in which failure may lead to injury, death, or major damage to property or the environment [7].

In order to gain confidence that safety-critical systems meet their safety obligations, these systems often need to undergo certification by a certification body. The goal of certification is to provide an assurance that a system has been deemed safe for use in a specific environment. The certification process is usually based on a specific standard applicable to the domain in which the system is operated, e.g., IEC61508 [14] for the certification of electrical, electronic or programmable electronic systems that are used in safety-critical environments, IEC61511 standard for the process industry [13], EN50129 [12] for railways, and NORSOK I-002 [21] for safety automation systems in the petroleum industry.

During the certification process, the system supplier needs to provide evidence demonstrating that the safety criteria envisaged by the underlying certification standard are being met. Since the evidence that is collected for certification depends to a large extent on the relevant certification standard, there should be a consistent interpretation of the standard being used, and all parties involved (including the supplier and certifier) should know what evidence is to be collected and maintained in readiness for certification. Without an explicit and agreed-upon interpretation of the underlying standard, divergent interpretations can (and commonly do) occur because of the standards being large documents that are expressed textually and in a language not easily understood by everyone. Redmill [32] mentions these issues in the context of IEC61508, where readers have difficulty understanding the standard and engineers are unable to interpret the standard consistently throughout an organization. Feldt et. al [9] find similar challenges in the space industry, where there have been problems between customers and suppliers due to the variance that exists in the interpretation of standards. Finally, Sannier et. al [35] highlight the gaps between the possible interpretations of the same standards in the nuclear energy industry. These earlier investigations all lend support to the need for having a common and formal interpretation of the requirements of a standard upon which certification is to be performed.

Further, when a standard is being used within an organization, the practices of the organization will need to be aligned with the standard, allowing the organization to
check which of its existing practices comply with the standard and which new practices need to be introduced and tailored. In order to achieve this alignment, suppliers need to relate the concepts of their application domain to the evidence requirements of the applicable standards. However, the majority of the evidence artifacts that the suppliers create and manage are based on the concepts for the application domain, and not those of the certification standards. Therefore, a systematic procedure is needed for creating the necessary evidence, such that the supplier can properly interpret the standard in the context of their application domain and verify whether sufficient evidence exists to satisfy all the requirements of the standard.

Finally, the format in which the evidence is presented for certification needs to be highly-structured in order to ensure that the evidence is readable and assessable. Traditionally, this has been very difficult to achieve via paper-based documents that form the basis of the certification evidence today. Thus, there is a case for managing this evidence electronically [5] in order to ease the navigation of the information and to allow for diversity of presentation, delivery and re-use.

The approach that we present in Section 3 uses MDE as the main vehicle for addressing the issues described above.

3. Approach

We propose an approach for assisting system suppliers with preparations for certification of their systems according to industry standards. Our solution makes use of UML profiles for specifying and automatically checking the constraints that must hold for compliance with safety standards. The solution takes into account the fact that the systems that need to be certified usually belong to a family (class) of systems, where each system is a variant of a base system. We consider the different levels of abstraction that are present and take advantage of the re-use that is possible in these types of systems to create a systematic and cost-effective solution.

The approach consists of four main steps as shown in Figure 1. Briefly, we start by creating a conceptual model of a certification standard (step 1). The resulting model is used for constructing a UML profile of the standard (step 2). We then apply the stereotypes of this profile to a domain model of the system that is undergoing certification (step 3). This step results in a precise link between the concepts in the certification standard and those in the system. Finally, we create instantiations of the (stereotyped) domain model for a specific certification of the system (step 4).

Steps 1 and 2 of our approach are performed once per standard. These steps require input from experts familiar with the certification process (including the standard used for certification), but not necessarily the application domain. Step 3 is performed once per application domain. Fulfilling step 3 requires expertise in both the application domain and the certification process. Step 4 is performed once for every individual system that is subject to safety certification. This step requires knowledge of the application domain, including the specific system(s) undergoing certification. In the remainder of this section, we present detailed descriptions of the four steps in our approach.
3.1. Step 1: Conceptual Model of a Safety Standard

In Section 2, we noted the need for having an explicit interpretation of the underlying safety standard. We achieve this in the first step of our approach through the creation of a conceptual model. A conceptual model is a formal description of some aspect of the physical and social world around us for the purpose of understanding and communicating amongst humans [20]. It employs some formal notation which is a combination of diagrammatic and linguistic constructs and serves as a point of common agreement amongst a team of people and can also be used as a means of forwarding this understanding to newcomers joining the team.

A conceptual model of a safety standard should thus capture the main concepts and relationships in the evidence information required for showing compliance to the standard. We use UML class diagrams [23] for conceptual modeling of safety standards. In UML class diagrams, 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, the package notation is used to make groupings of concepts and thus better manage complexity.

Our choice of UML is based on the fact that it is a well-recognized and standardized notation and that the UML class diagram notation adequately fulfills our needs. From a practical standpoint, it is in general useful to ensure that the notation being employed is already accepted in industry and at the same time easy to learn for practitioners.

Creating a conceptual model of a standard requires a careful analysis of the standard’s text to identify the salient concepts and relationships mentioned in the text. To record the concepts and relationships in a systematic way, we follow a process as we read through the standard: we label each concept with a name and create a definition for it in a glossary when it is first encountered. As we proceed through the text, we either create new labels or reuse previous ones based on the definitions we have. As we create the labels, we also identify the connections between them and represent all this within a UML class diagram. This process is in line with how qualitative data analysis [19, 6] is performed in general, whereby text is analyzed to describe, classify and connect the information presented in it.
We exemplify the above process over a small excerpt of the IEC61508 standard that concerns software safety life cycle requirements. The excerpt is shown in Figure 2. In the figure, we highlight the key concepts and relationships by enclosing the relevant text in a box and numbering it.

Box 1 shows that the concepts Phase and Activity are of importance during the software development lifecycle. Box 2 identifies some important relationships between phases and activities. An activity is performed during a phase and has specified inputs and outputs. Box 3 indicates that a generic life cycle is prescribed by the standard, though deviations in terms of phases and activities are not precluded. Box 4 includes the concepts: Technique, SafetyIntegrityLevel and TechniqueRecommendation - indicating that activities should Utilize certain techniques based on the safety integrity level. The same concepts and relationships can be found in several places in the standard and thus a glossary is created to ensure that consistent terms are used to refer to the same concepts and relationships. The glossary pertaining to the concepts shown in the excerpt can be seen in Table 1.

A graphical representation of the concepts and relationships from the excerpt is given in Figure 3. In the figure, we show some additional concepts (covered by the standard but not present in the excerpt) to aid the discussion about the process concepts in the rest of the article. We further note that the model in Figure 3 is still a partial representation of the concepts and relationships relevant to the development process. A full treatment can be found in Appendix A.

In Figure 3, we can see that an activity can include sub activities or it can be linked to another activity by either preceding or succeeding it; these relationships are modelled by the elements ActivityIncludes and ActivityLink, along with the properties
### Table 1: Glossary of Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activity</strong></td>
<td>A unit of behaviour in a process.</td>
</tr>
<tr>
<td><strong>Agent</strong></td>
<td>A person or organization that has the capability and responsibility for carrying out an activity.</td>
</tr>
<tr>
<td><strong>Artifact</strong></td>
<td>One of the many kinds of tangible by-products produced during the development of a system.</td>
</tr>
<tr>
<td><strong>Competence</strong></td>
<td>The ability to perform a specific task, action or function successfully.</td>
</tr>
<tr>
<td><strong>Individual</strong></td>
<td>Refers to a person.</td>
</tr>
<tr>
<td><strong>Issue</strong></td>
<td>A unit of work to accomplish an improvement in a system.</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>A social arrangement which pursues collective goals, which controls its own performance, and which has a boundary separating it from its environment.</td>
</tr>
<tr>
<td><strong>Phase</strong></td>
<td>A set of activities with determined inputs and output that are carried out at a specific time during the life of a system.</td>
</tr>
<tr>
<td><strong>SafetyIntegrityLevel</strong></td>
<td>The probability of a safety-related system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>An abstract concept that can represent a person, organization or standard that can be a source of requirements to a system.</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>A procedure used to accomplish a specific activity or task.</td>
</tr>
</tbody>
</table>

![Figure 3: IEC61508 Process Concepts and their Relationships](image-url)
Activities are to be performed by competent agents using techniques that are acceptable for the safety integrity level assigned to a component. All these aspects are modelled using the concepts Agent, Competence and Technique and the relationships Requires, CarriesOut, Possesses and TechniqueRecommendation. An activity may require certain artifacts as input and upon completion produce certain artifacts as output. These are modelled by the elements Artifact, InputTo, Output-From, Requires, Produces, Input and Output.

Sometimes a concept appears again, not in the same form as encountered previously, but rather as a specific case. In such instances, we use the generalization association from UML to indicate the relationship between the general and the more specific concepts. As an example, we show in Figure 4 a specific activity called SW-ModuleDesignDevelopment during which the design for software modules and their corresponding test specifications are created. It has specific input and output artifacts of different types. In Figure 4, we can see two types of artifacts – Specification and Instruction, which are specialization of Artifact. Subsequently, we have specializations of Instruction as ProgrammingGuidelines and DevelopmentToolsInstruction; and specializations of Specification as SWModuleDesign, SWModuleTest and SWSystemDesign.

The model resulting from the first step of our approach provides an explicit and precise interpretation of the evidence requirements in the underlying standard and is used in step 2 for creating a UML profile for the standard.

3.2. Step 2: Creating a UML Profile from a Conceptual Model of a Safety Standard

The conceptual model created in Step 1 forms the basis of a UML profile that we use to establish a link between the concepts in the system undergoing certification and the concepts in the certification standard being used. This link helps the supplier to verify that the collected evidence is in line with the requirements of the standard, and helps the certifier to better understand and assess the evidence artifacts provided by the supplier.

UML profiles [23] are a lightweight solution for extending the UML metamodel for use in a specific context (in our case, safety certification). They enable the expression of new concepts, notation and constraints by the introduction of context-specific
stereotypes. Moreover, to ensure that certain semantics are maintained in the models to which a profile is applied, one can add constraints to the stereotypes in the profile using the Object Constraint Language (OCL) [24]. The advantage of using OCL constraints is that they can be automatically checked using an OCL validation engine, thus providing a means of efficiently ensuring that the requisite evidence items are present in the model.

We create a UML profile for a standard by having the concepts in the conceptual model of the standard represented as extensions of the metaclass ‘Class’ in the UML metamodel. The attributes of each concept are represented as attributes of the class to which the stereotype is applied. Relationships between the concepts are mapped as extensions of the metaclass ‘Association’, and the properties of an association are mapped as extensions of the metaclass ‘Property’. Standard-specific and user-specific data types are captured as Enumerations. Grouping of model elements into packages is captured through extending the metaclass ‘Package’. Finally, for a given standard, a special stereotype extending the metaclass ‘Model’ is defined in order to initiate the guidance process.

The actual guidance for creating the required evidence artifacts is formulated in terms of OCL constraints and attached to the stereotypes. Specifically, we use OCL constraints for the following purposes:

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

As an example, we show in Figure 5 a fragment of the profile corresponding to the process concepts package discussed earlier along with the stereotypes that extend the metaclass ‘Model’ and ‘Package’. We further use this fragment to show examples of the five types of constraints mentioned above.

We note that, while the OCL constraints are attached to the profile stereotypes, the constraints need to be validated on the model elements to which the stereotypes are applied. Encapsulating the constraints fully within the profile is advantageous from a usability standpoint as the user is not exposed to the constraints and only has to validate them. On the other hand, such encapsulation comes at the cost of making the constraints more complex, because to bind themselves to model elements, the constraints will have to check which stereotypes have been applied to which model elements at validation time.

The first stereotype to be applied is to the model itself, in our example using IEC61508, this is the IEC61508Model stereotype, which starts the incremental guidance process about which types of evidence to create. This stereotype has attached to it OCL constraints that ensure that certain packages are present in the model to organize
the different model elements (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:

\[
\text{self.base\_Model.allOwnedElements}() \rightarrow \\
\exists e \mid e.\text{oclIsTypeOf}(\text{uml::Package}) \land \\
\text{not e.getAppliedStereotype('IEC61508Profile::HazardsAndRisks')}.\text{oclIsUndefined}()
\]

The keyword self refers to the element being constrained, in this case the IEC61508-Model 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, which would be the domain model in our case. The allOwnedElements is an operation that returns all the elements in the model. exists is an OCL operation that will check that at 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).

To model the development life-cycle, we have the stereotype Phase to model the different phases in a life-cycle and the stereotype Activity to model the activities. The Phase stereotype has a constraint attached to it that states that every phase must have at least one Activity defined for it, i.e., there must be elements that have the stereotype Activity in the same package and be connected to the element with the stereotype Phase. This is enforced through two different constraints, one on the class stereotype Phase and the other on the association stereotype PerformedIn (see Figure 3). On the stereotype Phase, we have a constraint (of type 3) that states that there should be an

---

1 For the sake of brevity, we have omitted the name and context of the constraints shown
association with a stereotype `PerformedIn` originating from the element that has this stereotype:

```java
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 a 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):

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

Constraints are also used to create properties for the elements on which stereotypes have been applied, or for creating user-defined types. For example, 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 combines constraints of types 4 and 5):

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

The above types of constraints allow the user to define domain-specific values for the enumerations (the profile only gives the name and type of the property).

The stereotypes together with the constraints defined on them are used in Step 3 to build a precise link between the concepts in the application domain and those in the standard’s domain, and to provide systematic guidance on how to create these links. We name this process elaboration and explain it in the next step.

### 3.3. Step 3: Elaborating a Domain Model for Compliance.

Once we have the UML profile created in step 2, we can proceed to apply the stereotypes of the profile to the elements of a domain model of the system to be certified. A domain model is a visual representation of real-world concepts and the relationships amongst them in a specific area of interest [17]. In the context of our work, we use the term domain model to refer to the concepts that represent the physical and abstract
A Christmas Tree and Well Head System has wellheads that attach to the sub-sea oil or gas wells and an assembly of control valves, pressure gauges, and chokes put on the top of a well to control the flow of oil and gas, known as a Christmas Tree (XT). The XT is housed on a manifold that is part of the Manifold and Jumper System. The Umbilical System is a housing that carries the power and communication line from the surface to the subsea equipment whereas the Riser System consists of all the equipment carrying the oil from the well to the surface. The Transmission and Communication systems are responsible for conveying the signals sent from the surface control equipment down to the sub-sea equipment. The Installation and Workover System is used to control and monitor sub-sea equipment during installation or maintenance. Finally, the Sub-sea Production Control System controls the valves and chokes on the XT and anywhere else on the manifold depending on the design of the system. This is done by sending and receiving data between the surface and the sub-sea equipment, thus allowing the engineers at the surface to monitor the sub-sea equipment.

We use sub-sea production systems to exemplify what constitutes a domain model in our context. In sub-sea production systems, there are a number of subsystems working together to extract the oil from the sea bed. In Figure 6, we show a simplified decomposition of sub-sea production systems into their constituent subsystems along with a brief description of the subsystems.

Within these large systems, we concern ourselves only with the Sub-sea Production Control System. We show a fragment of the control system and how it interacts with the other components in Figure 7. In this figure, we can see that the wellhead attaches to the XT (see the descriptions given earlier in Figure 6). The XT connects to the manifold that is anchored to the seabed via a structural frame called the template. Mounted on the XT is the Sub-sea Control Module (SCM) that receives commands from the Master Control Station (MCS) that is part of 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 MCS. 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 MCS. A more complete description of these components can be found in [15, 1].

The model in Figure 7 is a generic description of a class of systems — each variant of the system will have very specific types of oil wells, manifolds and sensors and actuators with specific actions that should take place in order to extract the oil. Following the norm in MDE, we assume domain models are represented as UML class diagrams [17]. We do not concern ourselves in this article with the construction of domain models. Good references and guidelines already exist [17].

To establish a mapping between a domain model and a standard, we apply stereotypes from the UML profile of the standard to the domain model. Specifically, the components of a class of systems in a particular application area (e.g., sub-sea production systems), the environment in which the system functions, and the key artifacts built throughout development.

Figure 6: Subsystems in a Sub-sea Production System
elaboration of the domain model 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 attached to 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).

As we stated in Section 3.2, our approach envisages a special stereotype extending the ‘Model’ metaclass for starting the guidance process. This stereotype is applied to the domain model itself, specifying what standard the domain model needs to comply with. We continue our exemplification using IEC61508. Thus, the IEC61508Model stereotype is applied to the domain model and the OCL constraints of this stereotype are validated. The violated constraints, shown in Figure 8, are the beginning of the guidance process for creating the evidence. The first requirement is the creation of four packages that have the stereotypes: Process, System, SafetyManagementSystem and HazardsAndRisks.

The rationale behind requiring these packages comes from the IEC61508 standard. The IEC61508 standard promotes 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 puts 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 specified within a safety life-cycle and kept in the Process package.

The packages themselves can be named using the supplier’s own terminology, but the specified stereotypes need to be applied. Once the package stereotypes have been applied, the next set of (violated) constraints will provide guidance on what stereotypes to apply next.

Figure 9 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.

First, we will show the application of the stereotypes that identify the system components. These stereotypes are shown in Figure 10. The stereotypes describe the basic elements needed to conceptualize safety-related control systems that involve both hardware and software. A Programmable Electronic System (PES) is represented by the stereotype ProgrammableElectronicSystem and is made up of one or more hardware blocks represented by the stereotype HardwareBlock and controlled by a number of software blocks – stereotype SoftwareBlock. A hardware block may represent a mechanical, electrical or electronic entity, both programmable and non-programmable, hence the existence of stereotypes NonProgrammableHardwareBlock and ProgrammableHardwareBlock.

By applying the stereotypes relevant to the IEC61508 standard, we are now relating the elements of the system as they pertain to the concepts described in the IEC61508 standard. The software of the MCS controls and monitors the sub-sea wells, so the stereotype SoftwareBlock is applied to this element. The system being controlled is the SCM which contains the Sub-sea Electronic Module (SEM), which in turn links to the different instruments. Thus, the stereotype ProgrammableElectronicSystem is applied to the SCM and the stereotype ProgrammableHardwareBlock to the rest of the elements. The applied stereotypes are shown in Figure 11.

If we now focus on the software and validate the MCS element, the validation will fail. In Figure 12, we show the violated constraints. All system blocks need to have a unique identification and this has not been added to the MCS element – in the petroleum industry, every component of a system has a unique identifier called a tag. The standard also recommends version control of the system components – a version
Figure 10: IEC61508 Profile fragment for System Stereotypes

Figure 11: Fragment of the Domain Model After Application of the System Stereotypes
attribute has not yet been added either, thus the violation of that constraint. A constraint on the SoftwareBlock stereotype requires one to show the decomposition level of the software, i.e. whether it is the entire software system or are we referring to a software module that is part of a system. The constraint specifies the name of the attribute ('Level') and the type ('Enumeration').

There are also various certification artifacts that need to be created during the construction of the software. These, among others, may be plans that guide the process of software construction (e.g., software verification plan), technical guidelines for the programmers such as programming guidelines or development tool instructions, and results of testing the software to show that it meets its requirements. All these are shown as required certification artifacts via the violation of the OCL constraints. In total, from Figure 12, we see that there are 25 different constraints that must be met for the certification of the software of the MCS.

As it would not be possible to show all the artifacts in a small legible diagram, we show in Figure 13 the resulting model after eight different artifacts have been added to satisfy some of the constraints. We have added an element to depict the software safety requirements (stereotyped with SWSafetyRequirements) that are created during the requirements analysis activity and a software validation plan (stereotyped with SWSafetyValidationPlan) which is the output of a safety validation planning activity. During the architecture design activity, the software architecture description is created (stereotyped with SWArchitectureDescription), as well as the software integration test specification (stereotyped with SWArchitectureIntegrationTest) and how to test the software with the hardware is detailed in the software/hardware integration

![Figure 12: Error Report Showing the Violated OCL Constraints for the Master Control Station](attachment:image.png)
test specification (stereotyped with SW_PE_IntegrationTest). The software system design activity results in the system design specification (stereotyped with SWSystemDesign) and the support tool and programming language selection activity results in the tool selection (stereotyped with DevelopmentToolsInstruction) and coding standards (stereotyped with ProgrammingGuidelines) to be used for development.

The unique identification, version and software level attributes have been added as well as a user-defined enumerated type for the 'SoftwareLevel'. 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' – these were the values that were most relevant for our industry partner. When the MCS element is validated after the addition of the new elements to the model, we see that the constraints related to the elements we have added are not violated any more and we can work on the remaining ones. We show the remaining violated constraints in Figure 14.

The new artifact elements that were linked to the MCS element would be defined in the process package. This was one of the packages that was required at the beginning of the elaboration process. The elements in the process package can be defined as needed, or be all created prior to their need and the links to the the MCS element established when the constraints fail. In Figure 15, we show some of the elements in the process package. Specifically, in the figure, we show the artifact elements that we have used in Figure 13. The artifact elements are shown on the left in blue. In the center we have the activities that lead to the creation of these artifacts, shown in orange. The development phase is software development, shown in green. For each activity, the standard recommends that the competence required should be documented as well as the agent who carries out the activity. We show an example of this for the SWSafetyRequirementsSpecification activity, coloured yellow.

We can see that the user can name the elements according to their own conventions, it is the stereotypes that are specific to the standard. For example, the software
Figure 14: Error Report Showing the Remaining Violated OCL Constraints for the Master Control Station

Figure 15: Fragment showing the Software Development Cycle according to IEC61508 Standard.
safety requirements element is named by the supplier as 'SoftwareRequirementsSpecification' and has the stereotype SWSafetyRequirements. It was created as an output from the activity called 'RequirementsAnalysis' and the analogous activity from the standard is identified by the stereotype as SWSafetyRequirementsSpecification.

In the above example, we presented very simply, how elaboration happens and new elements are added or existing ones are linked to new elements in order to satisfy the OCL constraints linked to the stereotypes. We have only shown the constraints for some of the elements, but all elements have such constraints and through this process of satisfying the OCL constraints, the domain models are updated to satisfy the requirements of the applicable safety standard. For ease of explanation, in the examples above, we show the validation of constraints per element, but they can be validated per diagram or per an entire model as well. Therefore, it is possible to work on the model in small increments while still being able to see what the overall state of the model is in regards to satisfying the requirements of the relevant standard. Once the domain model has been fully elaborated, it can be used for specific certifications of that class of systems as presented in step 4.

3.4. Step 4: Creating an Instance for a Specific Certification.

To support the certification of a specific system variant from a class of systems, the fourth and final step of the process 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 elaborated domain model is built to represent the specific properties of a system variant, and instances of the certification evidence are created as specified by the elaborated model. We recall that whereas steps one and two of our approach are performed once per standard, and step three once per class of systems (domain), the fourth step is performed once for each variant (actual system) that is subject to certification.

In Figure 16, we show an instance model conforming to the domain model that we presented in Figure 13. This instance model is partial in two respects: first, we do not show instantiations of all the elements in the model of Figure 13 and second, for those elements that we do instantiate, there is only one element instance. In an actual system, the number of instantiations per element can be more. For example, we show only one instance of each of the elements 'PressureTransmitter' and 'TemperatureTransmitter' whereas in an actual system there are numerous pressure and temperature transmitters. Also, for confidentiality reasons, we use sanitized names and do not give the real names of the element instances.

In the instance model shown, there is one template and one manifold. The manifold has two XT structures on it – X1 and X2, each connected to a wellhead. X1 has a temperature transmitter (TT1) attached to it and X2 has a pressure transmitter (PT1) attached to it. There are two SCMs, both controlled by a single MCS. The unique Id given to the MCS is 'T1823a', the version of the software for the MCS is 1.2 and this is the system level software version. These attributes were added to the MCS element due to constraints on the stereotype, allowing specific values to be set in the instance model. For the hardware equipment, the version would store the model number or serial number of the piece of equipment. An instance of the software system requirements
artifact called the ABC OilField Requirements is shown, as is an instance of the software system design artifact called the ABC OilField Software Design. Each of these artifacts has a URL linked to it so that that actual document can be accessed right from the tool environment. In this manner, the documents can be stored in any location and are simply accessed via a URL. This provides a mechanism that links diverse artifacts in diverse locations.

In this way, the evidence requirements for a specific system can be created in readiness for certification based on the relevant standard. Further requirements in terms of tool support for applying this approach are presented in Section 4 and the results of applying this approach in Section 5.1.

4. Tool Support

The tool support for our approach has to fulfill the following key requirements:

- Allow the creation of UML class diagrams which we use as a notation for representing our conceptual model.
- Allow the creation of a custom UML profile.
- Support the creation of OCL constraints at the level of the profile.
- Support the validation of OCL constraints.
- Provide customization of the messages given to the user when a constraint is violated.
- Provide the ability to create instances of the elaborated models.
- Provide the ability to create customized reports by querying the constructed models.

Any tool that fulfills the above requirements can be used as a platform for applying our approach. Amongst the existing alternatives, we have chosen Rational Software Architect (RSA) [11] by IBM to provide tool support for our approach. In addition to
meeting all the above requirements, RSA is a mature and industry-strength tool with good usability, thus making it easier to apply our approach in an industrial setting and making it more likely for the approach to be adopted by practitioners.

We have successfully used RSA version 8.03 in our case study to support all the steps of our approach described in Section 3. Specifically, we used RSA to create the UML class diagrams for the conceptual model of the standard as well as the domain models for elaboration. We then used RSA to create the UML profile of the IEC61508 standard. RSA supports adding OCL constraints at the level of the profile. More importantly, it has a built-in OCL validation engine that we could utilize to provide the guidance for elaborating the domain models according to the IEC61508 profile. The messages given to the user when a constraint is violated can be customized. RSA also includes a report designer based on Business Intelligence Reporting Tool (BIRT) [8], that can be used to publish reports in user-defined layouts based on the data in the models. While we have not yet customized this report designer for generating safety certification reports, the existence of such a flexible report generation framework was an important consideration that we had to account for.

The domain models can be created in a hierarchy. This allows one to start with a high-level view and then create more detailed models as and when necessary. Large diagrams can also be split into a number of smaller diagrams, but if an overall view of a particular element is required, then a 'browse' diagram can be automatically generated. A 'browse' diagram shows all the elements that a chosen element is related to and helps in understanding how that element fits into the overall system depicted in the model providing a snapshot of the overall context of an element. These diagrams are not permanent diagrams: they are generated from the most current information in the models and hence a browse diagram can be refreshed to show the latest state of the model elements. It is also possible to convert a browse diagram to an editable diagram. This provides a means to both get an overall context of an element and proceed to edit it if necessary.

RSA further allows for custom documentation to be added to the stereotypes. When the mouse cursor hovers over a stereotype, a pop-up window displays the associated documentation, as shown in Figure 17. All stereotypes can be documented in this way to provide further assistance to the user while applying the stereotypes.

To help in the creation of the instance models, there is a properties view that shows the slots for the selected instance. Each slot is a mapping to an attribute of the classifier that has been instantiated and every time a value is created for a particular attribute, the properties view is updated to reflect the change. The creator of the instance model can thus see which slots have values already and which ones still need values. In this way, RSA can guide the user in creating a complete instance model.

5. Evaluation

In this section, we first report on an industrial case study performed in the maritime and energy domain (Section 5.1), followed by a description of a survey performed among domain experts to better understand their perceptions about our approach (Section 5.2). The case study enables us to determine the feasibility of our approach to
support certification and the survey helps assess whether domain experts see benefit in adopting our approach in a real industrial context.

5.1. Case Study

Our case study is aimed at investigating the feasibility of our approach and the level of effort involved in its application for this case study. Below, we provide a detailed description of the context, execution, and outcomes of the case study.

5.1.1. Nature of the Case Study

The subject of the case study is a new approach for improving upon the current practice of safety certification. Our case study can therefore be seen as an improvement case study as described by Runeson et. al. [34]. An important characteristic of our case study, like many other case studies that are conducted in industrial settings, is that there is very limited control over the industrial process that is the subject of investigation, in our case, the certification process. This in particular meant that we could not set up a comparative study and twice run the same certification process. Comparisons with current practice (or new candidate approaches) therefore have to rely on the stakeholders’ perceptions. We will discuss this in more depth in Section 5.2.

5.1.2. Research Questions

The case study is targeted at answering the following research questions:

- **RQ1. Is the approach feasible?** More specifically, this question is concerned with (1) whether it is possible to represent the evidence requirements of a safety standard in terms of a conceptual model, and (2) provided that the answer to the first part is positive, whether it is possible to encapsulate the guidance for the creation of certification evidence into a UML profile based upon a standard’s conceptual model. For answering RQ1, we do not concern ourselves with the creation of the domain models and instance models envisaged in our approach. These activities do have implication on the effort involved in carrying out the
case study (see RQ2) but are technically well-understood in practice and do not require a feasibility study.

- **RQ2. Is the effort involved in the application of the approach acceptable?**
  The answer to this question is based on the level of effort spent throughout the case study. Effort is an important factor for the successful adoption of a new approach. If practitioners do not find the level of effort reasonable, they are unlikely to adopt the approach.

### 5.1.3. Case Selection

Our approach was motivated by the issues that our industrial partners faced during safety certification, both on the side of the certifiers and the suppliers of safety-critical systems. To apply the approach in an industrial setting, we needed two prerequisites to be in place: (1) access to a safety-critical system that has undergone safety certification recently, is currently being certified, or is about to be certified in the near future. Old certification projects were deemed unsuitable for a case study due to the difficulty of acquiring sufficient details about them. (2) access to domain experts and securing adequate participation from them for the case study. We note that safety certification is a necessary but relatively infrequent event: entirely new safety-critical systems that need be certified are rare and the existing systems evolve rather slowly and require re-certification once every few years. Due to the scarcity of cases, we had to be opportunistic with case selection, as long as the two prerequisites above were satisfied.

The system suppliers that we had access to were involved with the certification of sub-sea production systems (discussed earlier in Section 3). The timing of our case study coincided with the construction of a new oil field, whose sub-sea production system needed to be certified in the near future. Within this system, we still had to choose a specific part to work on as these systems are very large and performing a case study on a complete system would have required resources beyond what was available at the time. Since we are primarily interested in software safety certification, we chose to concentrate our study on the software aspects of the sub-sea production control system for the new oil field. More specifically, the goal of the case study was to determine whether the software development plan being used by the supplier complied with the certification requirements. The software development plan outlines the activities that are carried out during software development and the resulting artifacts that are used as evidence during certification to show compliance. Our aim was to help the supplier determine which artifacts to create during the development of the software for the new control system.

The certification standard that the supplier needed to comply with was the IEC61508 standard [14] which sets forth the certification requirements for control systems that incorporate both mechanical and electronic components controlled by software. The aim of the standard is to ensure that safety-critical systems operate correctly in response to their inputs and that the system is brought to a safe state should a hazardous situation occur – known as functional safety. This standard is a large and comprehensive generic standard that is utilized in many domains, making it a good indicator for the feasibility of our approach. Moreover, it has been specialized for a number of domains such as the process industry [13], railways [12], automobiles [16] and others. In this sense,
being able to apply our approach successfully for this standard is a good indicator of the generalizability of our work.

5.1.4. Data Collection Procedure

The procedure taken for conducting the case study closely followed the approach described in Section 3. In the first step, a conceptual model was built for the IEC61508 standard by analyzing the text of the standard. This work was done by the first two authors. The model was subsequently reviewed by an expert specializing in certification based on IEC61508 and subsequently was revised. The revised model was then presented to a group of twenty-eight certification experts in an industry workshop. During this workshop, the modelling notation was explained and the model itself fully presented. A question and answer session was held. This session resulted in no further changes to the model.

The second step was the creation of the IEC61508 profile. This was carried out by the first author. The basis for the profile is the conceptual model of IEC61508 built in the first step. In addition, OCL constraints were added to the profile, to provide guidance in elaborating the domain model.

The third step was the elaboration of the domain model. Since we did not have a domain model for sub-sea control systems a priori, we had to develop one. To this end, we first created a generic domain model for these systems using a general description of the systems found in [1, 15]. We then reviewed and refined this model over several meetings with a domain expert in the sub-sea domain at the supplier company where we were conducting our case study. The resulting domain model is one specialized to the needs of the supplier and includes concepts specific to the sub-sea control systems developed by the supplier. Once the domain model had been created, the elaboration process was carried out. As mentioned earlier in Section 5.1.3, a complete elaboration was not possible with the resources available to the supplier, hence the elaboration was carried out for the parts that were most relevant to the system supplier: the software development process – the activities and artifacts mentioned in the software development plan used by the supplier were modelled and the elaboration process carried out to check if these were sufficient or whether other activities and artifacts needed to be added. The fourth and final step was the creation of an instance for the particular system that was being certified. Steps 3 and 4 were carried out by the first author and the results reviewed by the experts at the partner company.

5.1.5. Results

In this section, we present the results of our case study. Parts of the case study were used to explain the approach in Section 3. In this section we concentrate on providing an overview of the outcomes of the case study without repeating any technical details that have already been discussed in Section 3 along with examples from the case study.

Step 1 (Conceptual Model of IEC61508). The IEC61508 standard consists of seven parts of which parts one, two and three contain the requirements for the functional safety of the system. Each of these parts describes an overall safety life-cycle to achieve the required level of safety; part one for the overall system, part two for the hardware components, and part three for the software components. Parts four, five
In Table 2, we provide some statistics about the IEC61508 conceptual modeling activity and the contents of the resulting model. The standard is expressed as numbered requirements. A requirement in this case is a numbered item that expresses some criteria that must be met for a system to comply with the standard. We examined 318 such textual requirements and extracted 95 concepts of importance that are linked together by 51 relationships. We grouped these concepts into 10 packages encompassing related concepts. The effort involved in carrying out this activity is shown in Table 6 and discussed in Section 5.1.6, under research question RQ2. This conceptual model of the IEC61508 standard is the basis of the profile described in the next section.

### Step 2 (UML Profile of IEC61508)

In this step, all the concepts and relationships from the conceptual model of step 1 were transformed into stereotypes in a profile. We then augmented the profile with OCL constraints to provide guidance for elaborating the domain models. In Table 3, we provide a summary of the contents of the resulting UML profile. The number of stereotypes that extend the metaclass `Class` is the same as the total number of concepts in the conceptual model and the number of stereotypes that extend metaclass `Association` is the same as the number of associations in the conceptual model. We then have stereotypes extending the metaclasses `Model` and `Package` for helping to organize the model elements and finally their are OCL constraints for elaboration; recall from Section 3.2 that we have five different types of constraints, we show the number of each type in Table 3. This profile is then used for the elaboration process described in the next section.

### Step 3 (Elaborating the Domain Model of Sub-sea Production Systems)

The domain model of the system was made 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 [15, 1, 21, 36]. A fragment of the high-level break-
Table 3: Summary of Profile Stereotypes for IEC61508

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of stereotype extending metaclass Model</td>
<td>1</td>
</tr>
<tr>
<td>Number of stereotype extending metaclass Package</td>
<td>4</td>
</tr>
<tr>
<td>Number of stereotype extending metaclass Class</td>
<td>95</td>
</tr>
<tr>
<td>Number of stereotype extending metaclass Association</td>
<td>51</td>
</tr>
<tr>
<td>Total number of OCL constraints in profile</td>
<td>218</td>
</tr>
<tr>
<td>Number of Type 1 OCL constraints in profile</td>
<td>95</td>
</tr>
<tr>
<td>Number of Type 2 OCL constraints in profile</td>
<td>53</td>
</tr>
<tr>
<td>Number of Type 3 OCL constraints in profile</td>
<td>42</td>
</tr>
<tr>
<td>Number of Type 4 OCL constraints in profile</td>
<td>20</td>
</tr>
<tr>
<td>Number of Type 5 OCL constraints in profile</td>
<td>8</td>
</tr>
</tbody>
</table>

down of sub-sea production systems was shown in Figure 6. The complete high-level model is shown in Figure B.45 in Appendix B. After creation of the high-level domain model, we then concentrated on modeling the sub-sea production control system. We modelled only a small part of this system resulting in a model with forty-six elements and sixty-two relationships in it, of which twelve of the elements were subsystems requiring further breakdown but were not in the scope of the case-study. We do not show this model here as it contains proprietary information from our supplier, however, we did present a small sanitized fragment of it in Figure 7 (Section 3.3).

The elaboration of the domain model began with applying the stereotype IEC61508Model to the domain model. We then continued with applying the system stereotypes and subsequently concentrated on the software control system to determine whether the activities and artifacts mentioned in the software development plan used by the supplier were in line with those prescribed in the standard. The process was as described in Section 3.3. We added all the artifacts that were needed to satisfy the constraints for the software.

When artifacts are added to a model, their constraints will require the addition of the activities that generate those artifacts. This led to the creation of the complete process model for the software development. Once all the constraints were satisfied for the software, we compared the activities and artifacts created during the elaborated model with those that the supplier had as part of their software development plan. The supplier was keen to check how well they satisfied the requirements of the IEC61508 standard based on their current software development plan. We show a summary of this exercise in Table 4. The standard defines 16 activities that are carried out for software development and the requisite artifacts that are the output of these activities. From the 16 activities, the supplier had 10 of them defined in their software development plan and only 10 of the required artifacts were in the plan from the requisite 27. The supplier needed to update their software development plan and add the missing activities, artifacts and required competence and disseminate this information to their engineers.

**Step 4 (Instantiating the Domain Model).** The instance model was created for the
Table 4: Supplier Software Development Plan Versus IEC61508 Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of activities required by IEC61508 for software development</td>
<td>16</td>
</tr>
<tr>
<td>Number of activities in the supplier software development plan</td>
<td>10</td>
</tr>
<tr>
<td>Number of artifacts required by IEC61508 for software development</td>
<td>27</td>
</tr>
<tr>
<td>Number of artifacts in the supplier software development plan</td>
<td>10</td>
</tr>
<tr>
<td>Number of activities requiring definition of competence in the supplier’s software development plan</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 5: Summary of Supplier Software Development Plan at the Partner Company

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of activities to be carried out for software development</td>
<td>16</td>
</tr>
<tr>
<td>Number of artifacts to be produced during software development</td>
<td>27</td>
</tr>
</tbody>
</table>

software development process of a specific system. The model represented all the activities that would be carried out and the artifacts generated. It also highlighted that the developer needed to define the competence required for some of their activities. The instance diagram created for the software development is quite large as it defines all the activities for software development and the artifacts generated as evidence from these activities. Some of the activities are carried out multiple times, e.g., the module design activity will be performed for every module in the software, similarly with the module testing activity. Thus there will be as many artifacts specifying the module design, the module test specification and the results of testing, as the number of modules in the system (we cannot show this information due to confidentiality reasons). All this information is captured in a central place and linked to the actual artifacts that will be necessary for certification (as shown in Figure 16 in Section 3.3). The numbers in Table 5 show the different types of activities to be carried out and the analogous number of artifacts. The actual number for a system will vary depending on how the system is broken into subsystems and modules.

5.1.6. Discussion

Below, we discuss the results of the case study focusing on answering the research questions that we presented in Section 5.1.2.

- RQ1. Is the approach feasible? We could successfully extract concepts and relationships from the IEC61508 standard based on the process described in Section 3.1. After finishing the process, we amalgamated the different diagrams into one main diagram that showed all the concepts and their relationships for the overall standard. For software development, we kept separate diagrams per software activity as shown in Appendix A.

While identifying the concepts, we tried as much as possible to use the terminology used by IEC61508. Specifically, we were trying to use the terminology
defined in part four of the standard, but we found several terms that were used in the standard but not defined. For example, the terms Activity, Competence, Enhancement and Issue all appear in the text of the standard but no definition is given for them, the assumption perhaps being that these are naturally understandable to readers. However, providing a definition for each concept is an important prerequisite for creating an explicit interpretation of the standard and minimizing the possibility of ambiguity. For the terms used but not defined by the standard, we had to develop our own definitions based on both the context of their use and the definitions found in the literature on safety and reliability.

Providing definitions for the relationships between concepts turned out to be a more challenging task, as the standard does not explicitly discuss the links between the concepts. The names for the relationships were chosen based on our reading of the text; however, we had to develop the definitions for the relationships on our own in the same manner as for the concepts that were not defined by the standard.

After revisions based on comments from the certification expert, we presented the model at a workshop where the participating certification experts agreed that we had captured the salient concepts and relationships within the standard. An important issue to note here is that our model should not be viewed as a comprehensive interpretation of IEC61508. In particular, the standard contains numerous requirements concerned with how to perform the various lifecycle activities and how to describe the contents of the artifact, e.g., "Coding standards shall be a) reviewed as fit for purpose by the assessor; and b) used for the development of all safety-related software", and "The documentation shall be easy to understand by those having to make use of it". We do not capture such requirements in our conceptual model as we are concerned with the type of evidence required for certification and not the quality attributes of the evidence. At the most we can add this text to the documentation for the stereotype concerned, thus the user will see it when using the stereotypes and be aware of these requirements.

Overall, we observed that having a graphical representation of the conceptual model was very useful for presenting it. The certification experts and practitioners from the supplier agreed that the choice of UML class diagram for representing the conceptual model was useful and easy enough to understand. This finding is confirmed by the survey that we present in Section 5.2.

With regards to the construction of the profile for IEC61508, it was straightforward to translate the concepts and their relationships from the conceptual model into stereotypes. The challenging part was developing the OCL constraints. The constraints were not only used to establish the properties of the stereotypes themselves but to add additional information that would guide the user in applying the profile to domain models. All these constraints were added manually to the model. From the five types of constraints mentioned in Section 3.2, it is possible to automate the process of adding the constraints of type 2, 3, 4 and 5 as these are derived directly from the conceptual model based on the type of metaclass being stereotyped. This automation can take advantage of the fact that models
can be queried and then software can be written to extract the required information and create the profile along with the OCL constraints. However, we chose not to pursue this route as this would have involved solving technical challenges as opposed to solving a research problem. Instead, we were more interested in assessing whether guidance could be provided for elaboration using OCL constraints. Constraints of type 1 were added for the software development process and required thought into what types of constraints would provide adequate guidance and which stereotypes the constraints should be added. Here, automation would not be possible as for types 2–5 since the constraints are not extracted directly from the conceptual model.

With regards to the domain model elaboration, we found the profile to be effective in guiding the construction of a relationship between the standard and the application domain. For the purpose of showing feasibility, we focused on the elaboration of the artifacts created during software development. Even with this limited focus, the supplier viewed this as a useful exercise, as they could more easily see which activities and artifacts were not part of their current software development plan. Some activities were missing even though they were carried out but their results were never explicitly documented for certification – this was the case for module design, which was carried out by the software engineers as part of the implementation work and never explicitly documented. Other activities were missing because they were not carried out as part of software development per se but rather as management activities – such as planning for the safety assessment of software. Thus, elaboration helped the supplier identify the diverse pieces of information that were relevant to software but were organized under headings other than software. All this information could now be linked to the software development plan and presented as such to the certification body at certification time. Furthermore, the supplier realized that they needed to link the competence data that they keep about their employees to the activities of the developments process so as to show that the work was being performed by people (referred to as agents in Table 4) with the right competence. This information was also available but never linked to the certification information.

We note that our feasibility argument for domain model elaboration, as given above, focuses on a small fragment of the domain model, and in this case the number of constraints that are violated is manageable. However, we acknowledge that for a complete model there could potentially be a large number of violations that would need to be dealt with. In this case, it would be useful to have some form of prioritization of which constraints should be tackled first and which later. We do not propose a solution for this issue in this article and leave it for future work.

In summary, we recall that our focus for showing feasibility was the construction of the conceptual model of a standard and its use in creating a UML profile to provide guidance for the creation of certification evidence. To this end, we have shown that both aspects are feasible in a realistic certification context. As we would like our approach to be adopted in industry, we cannot rely on feasibility alone: we need to also consider the effort required and that is the subject of RQ2,
Table 6: Effort Involved in Creating the Conceptual Model for IEC61508

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>First pass through the entire standard</td>
<td>6 weeks</td>
</tr>
<tr>
<td>Second pass through the standard, involving:</td>
<td></td>
</tr>
<tr>
<td>– Extraction of concepts and relationships</td>
<td>14 weeks</td>
</tr>
<tr>
<td>– Definition of concepts and relationships</td>
<td></td>
</tr>
<tr>
<td>– Choosing appropriate modelling constructs</td>
<td></td>
</tr>
<tr>
<td>– Graphically representing the concepts and relationships</td>
<td></td>
</tr>
<tr>
<td>Review by certification expert</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Application of comments provided by certification expert</td>
<td>2 weeks</td>
</tr>
<tr>
<td>IEC61508 workshop</td>
<td>1 Day</td>
</tr>
</tbody>
</table>

• RQ2. Is the effort involved in the application of the approach acceptable?  
The creation and refinement of the conceptual model required approximately 6 person months. A breakdown of the effort required for the various tasks performed is presented in Table 6. The bulk of the effort was spent on extracting, defining and modelling the concepts and relationships in the standard. The rest of the time was mainly used, first, to familiarize ourselves with the IEC61508 standard, which is a substantial document in itself; and second to refine the created model.

The creation of the profile took approximately 3 person months. This includes the (one-time) effort taken to investigate how we could use OCL constraints for guiding domain model elaboration and implement the constraints in the RSA tool (discussed earlier in Section 3.2).

The construction of the domain model took another 2 person months. Most of this time was spent understanding the domain and creating the domain model. We had access to domain experts but they were not available to create the models, only to check them and provide feedback. Thus, the first author spent time learning the domain of sub-sea control systems and reading the system documentation provided by the supplier in order to create the models which were then refined based on comments from the experts.

The process of elaboration and instantiation were both completed within one month. This is comparatively less effort than the other tasks and was partly due to the limited scope in steps 3 and 4. It should be noted though, that for the little effort that was spent on these steps, the supplier was very pleased with the findings that the elaboration gave rise to.

The largest level of effort in our study was spent on the creation of a conceptual model for IEC61508 (the underlying certification standard). We anticipate such conceptual modeling to require a sizable effort for other standards as well; however, this activity needs to be performed only once per standard, or standard
revision. We expect the effort for the creation of a profile to be less on future applications, as we had to address several technical details in relation to using OCL constraints in our first application.

The next most effort-consuming task was the creation the domain model of the system which was done by a researcher unfamiliar with the domain or its terminology. The effort may be reduced if the models are created by someone familiar with the domain. Note that this effort is only needed once for a domain.

The general criterion used by the experts to decide whether the level of effort was acceptable was that the (perceived) cost savings resulting from the approach should reasonably exceed the cost of applying the approach. The experts took several factors into account in their estimations. These included the current time span for the certification of a typical safety critical component (between 8 months to over a year), the effort that had to be invested into the preparation of certification documents and attending certification meetings, the costs associated with the involvement of an external certification body, and the side benefits that the models built in our approach could bring, e.g., for training of new staff members.

The experts felt that the initial effort for creating the conceptual model of the standard and the profile was somewhat high; however, since this is a one-time task, they found the level of effort to be acceptable. The justification here was that, given the number of projects that the supplier company would have to certify using the IEC61508 standard in the future, this effort would pay for itself over time. The creation of the domain model was a useful exercise to the supplier as well. This effort too can be spread over a number of projects, as the same model is used for all sub-sea production systems. Thus including a high amount of re-use in the steps of the approach has been a useful choice.

5.2. Survey

Following the completion of our case study, we conducted a workshop where we presented our solution and collected feedback from practitioners through a survey. Broadly, the aim of the survey was to investigate the likelihood of our approach being adopted by practitioners in industry. This investigation is based on the factors involved in technology adoption. By examining these factors we have created a questionnaire that allowed us to compile the perceptions of practitioners regarding our approach. In this section, we discuss the design and results of this questionnaire.

5.2.1. Data Requirements

While preparing for the survey, we had to consider what factors would be of interest to the practitioners in assessing whether to adopt a technology, as well as what we needed to ask our participants in order to obtain useful feedback. We found the factors in Rogers’ theory of innovation diffusion [33] highly relevant to consider:

- **Trialability** is the degree to which the technology can be tried on a limited basis or adopted in increments.

- **Compatibility** is the degree to which the technology is perceived to be consistent with existing values, experiences and needs of the practitioners.
- **Relative advantage** is the degree to which the new technology is perceived to be better than what is currently used.

- **Observability** is the degree to which the results of using the technology would be visible to others. This is important as visibility kindles discussion amongst peer groups and helps the spread of the technology. In our case, we would want visibility to the other members of the team and the certification body as well. This would then help spread the use of the technology.

- **Complexity** is the degree to which a technology is perceived to be difficult to understand or use.

All the above factors are based upon the perceptions of the practitioners. There may be ways to objectively measure the relative advantage of a technology or its compatibility and so forth but it is the perceived advantage in terms of a particular factor that matters most for the adoption of the technology.

The case study described in Section 5.1 enabled us to try our approach in a small but realistic setting, providing a suitable context to examine trialability. With regards to compatibility, we knew beforehand that our solution presented something different from what the practitioners were accustomed to. While the solution was developed in response to the practitioners’ direct needs, we have not yet attempted to integrate it into the current workflow of certification activities at the partner company. Therefore, we cannot at the moment address compatibility in a direct way. The same is true for relative advantage – we did not want to compare the current mode of working with our solution as we did not have any means to show comparison. Instead of asking direct questions about compatibility and relative advantage, we decided to ask a question about whether the experts would see value in adopting our approach.

As for observability, we note that the approach is collaborative by design and would entail an evolution of work practices, thus contributing to visibility within the entire team. In fact, the execution and the outcomes of our case study although limited in scope, already attracted considerable attention at the partner supplier. Further and in relation to observability, we wanted to investigate if the partner supplier would be willing to use the developed models for interaction with the certification body as well, in turn increasing the observability of the approach. We have a question to this end in the survey. Finally, our survey has questions to directly assess the perceptions of the participants about the complexity of the approach.

### 5.2.2. Data Generation Method

To carry out the survey, we needed to first provide a reasonably thorough overview of our solution to the experts. This was done in an interactive workshop lasting for two hours. We wanted to present our work to as many relevant practitioners at the supplier company as possible. Our sampling frame was practitioners involved with developing software for safety critical systems that are to be certified by a third-party certification body. Within this sampling frame, we used the snowball sampling technique [22] combined with purposive sampling [22] to select the participants for the survey. In snowball sampling, one person is picked from the target sample frame, and once data has been gathered from this person, she is asked to recommend others to contact for
further data collection. This process can be repeated to add further data samples. In purposive sampling, the sample group is hand-picked to include those who would likely produce the most valuable data for the purpose of the research. In our case, we had a champion who had been working with us throughout the case study and was familiar with both the purpose and technical details of the work. We asked her to contact all staff members she knew of, who satisfied our sampling frame criterion. The champion sent out an email invitation to twenty-three staff members including developers, project managers and product managers. The email informed the invitees that a model-driven approach for IEC61508 certification would be presented. After this, neither we nor our champion had any control over the sample – we surveyed those who came to the workshop.

During the workshop, we first explained all the steps of our approach, providing examples drawn from the partner company’s application domain. We then presented the results of our case study, followed by a demonstration of tool support. There was time for questions so that we could clarify any concerns. After the question and answer session, we circulated a questionnaire for the participants to answer. The questionnaire was anonymous, although participants were given the option to provide their name and contact information in case any follow-up was required – the choice was up to them to participate. Due to the strict time constraints that we had to observe, we had to be very selective with the questions that we put on the questionnaire – we needed to cover our data requirements while ensuring that the questionnaire was clear and succinct.

5.2.3. The Questionnaire

The questionnaire was divided into four parts. The first section, Q1–Q2 shown below, was about the background of the subjects related to certification in general. We explained in the questionnaire that “certification-related experience” covers the following: (1) Attending tutorials and workshops on certification processes and standards; (2) Self-reading of certification standards; (3) Attending certification meetings; (4) Constructing and reviewing certification reports and deliverables.

| Q1. Is certification an important aspect of your job? |  |
| □ Yes |  |
| □ No |  |

| Q2. How much experience do you have with certification-related activities? |  |
| □ Less than 6 Months |  |
| □ More than 6 months but less than 12 months. |  |
| □ More than 1 year but less than 2 years |  |
| □ More than 2 years |  |

The second section, Q3–Q6, was about the participants’ experience with the IEC61508 standard for certification. Although we had based our work on prior knowledge about the difficulty in using text-based standards, we wanted to ensure that this was also the case in this group. For this set of questions, if the answer to Q3 was “No”, the participant was to skip Q4–Q6.
Q3. Is your own work (current or past) related to demonstrating compliance to IEC61508?

☐ Yes
☐ No

Q4. Have you read the IEC61508 standard?

☐ Entirely
☐ To a Great Extent
☐ Somewhat
☐ Very Little
☐ Not At All

Q5. Based on your experience, how easy to understand is the text of the IEC61508 standard?

☐ Very Easy
☐ Easy
☐ Average
☐ Difficult
☐ Very Difficult

Q6. Based on your experience, is the IEC61508 standard easy to use for certification?

☐ Always
☐ Usually
☐ About half the time
☐ Seldom
☐ Never

The third section, Q7–Q8, was about the modelling of safety standards. In particular, we wanted to find out what the participants thought about modelling of a textual standard.

Q7. Was the presented conceptual model easy to understand?

☐ Very Easy
☐ Easy
☐ Average
☐ Difficult
☐ Very Difficult

Q8. If given a conceptual model of a standard like the one we presented, would you use that model to help in understanding the standard?

☐ Definitely
☐ Very Probably
☐ Probably
☐ Possibly
☐ Probably Not
☐ Very Probably Not

The final section, Q9–Q12, was aimed at obtaining feedback about the overall approach and using model-driven engineering. For confidentiality reasons, we do not
reveal the name of the collaborating company and refer to it as “Company A” in Q10 and Q12.

**Q9. How easy to follow were the steps in our approach?**
- [ ] Very Easy
- [ ] Easy
- [ ] Average
- [ ] Difficult
- [ ] Very Difficult

**Q10. Would you see value in adopting the presented approach at Company A for certification?**
- [ ] Definitely
- [ ] Very Probably
- [ ] Probably
- [ ] Possibly
- [ ] Probably Not
- [ ] Very Probably Not

**Q11. Do you find the models simple enough to use for communication with a certification body?**
- [ ] Definitely
- [ ] Very Probably
- [ ] Probably
- [ ] Possibly
- [ ] Probably Not
- [ ] Very Probably Not

**Q12. Does the presented tool provide useful assistance for certification at Company A?**
- [ ] Definitely
- [ ] Very Probably
- [ ] Probably
- [ ] Possibly
- [ ] Probably Not
- [ ] Very Probably Not

5.2.4. Survey Results

Out of the twenty-three invitees, twelve attended the workshop, yielding a response rate of approximately 52%. All groups (i.e., developers, project managers and product managers) were represented by those in attendance.

Based on the responses obtained, certification was an important aspect of the job for all but one participant (Figure 18). This participant was a consultant, working as a project manager who would be engaged in certification activities in the future and hence had chosen to attend the workshop. Overall, 58% of the participants had over two years of certification experience and a further 17% had at least one year of experience with certification (Figure 19).
Regarding the experience of the participants with the IEC61508 standard, all except two had worked with this standard (Figure 20). From the two that had not worked with the standard, one was the consultant mentioned earlier while the other had experience with certification using a standard other than IEC61508. Moreover, certification was an important aspect of his work and he had attended the workshop because he was interested in the application of model-driven engineering to other standards.

From the set of participants who had used the IEC61508 standard, 40% had read the entire standard and a further 20% had read almost the entire standard. The rest had read some parts of it (Figure 21). Half of these participants felt the standard was difficult to understand and the rest thought it was of average difficulty (Figure 22). With regards to ease of use, 40% of the participants who had read the standard indicated that IEC61508 is seldom easy to use for certification, while 30% found it easy half the time; only 30% agreed that the standard is usually easy to use (Figure 23). When the findings shown in Figures 20 and 23 are taken together, we can conclude that 83% of the participants need to use the standard as part of their work and yet only 30% of them found the standard easy to use. This lends support to further research targeted at making large standards like IEC61508 easier to use for practitioners. Note that for Q5 and Q6 (Figures 22 and 23), we presented results from the subset of participants that had used the IEC61508 standard for demonstrating compliance, as determined by Q4 (Figure 21). The remainder of the results are from the entire group of participants.
When presented with the conceptual model of the IEC61508 standard, 8% of the participants found it very easy to understand and a further 67% found it easy (Figure 24). A further 17% thought the conceptual model was averagely easy to understand and 8% found it difficult. If we compare this to the ease of understanding of the textual standard, we find that 92% of the participants found the conceptual model to be very easy, easy, or averagely easy to understand compared to 50% finding the textual standard averagely easy while the other 50% finding it difficult to understand. Figure 25 shows that the participants unanimously agreed that they would probably use the model to help them better understand the textual standard. This shows that creating a conceptual model of a standard can be useful even when not employed as part of a wider certification strategy.

The final four questions concern our approach as a whole. As stated earlier when explaining the questionnaire, we cannot disclose the name of our industry partner and refer to it as “Company A”. We need to note though that the industry partner is a large supplier of safety-critical systems, with many of its systems subject to certification standards, IEC61508 being one of the key ones. Thus, they were a good representative company for evaluating our approach.

We found that 8% of the participants perceived the approach as being very easy to follow, while 67% thought it was easy, and a further 17% thought it was averagely
easy to follow; no one found the approach difficult to follow (Figure 26). In terms of adoption, 42% of the participants thought that there was definitely value in adopting the approach and a further 50% thought the approach was very probably worth adopting. Thus, 92% of the participants were in favour of adopting the approach. The remaining 8% were not negative either and thought the approach was probably worth adopting as well (Figure 27). All participants agreed that the models created during the application of a model-driven certification approach were simple enough to use in communication with the certification body (Figure 28). The current implementation of the approach using IBM Rational Software Architect [11] was also thought to be useful: 33% of the participants believed that the tool would definitely be useful for certification and a further 42% thought it would very probably be useful. The remaining 25% of the participants were spread between probably useful (17%) and possibly useful (Figure 29). No negative opinions were expressed about the tool.

In summary, the answers suggest that the approach was overall viewed to be easy to understand and the participants thought that it would be advantageous to use it within their context. This response indicates that model-based approaches to certification should be explored further.
6. Related Work

There are two main areas of work related to the approach presented in this article: managing certification evidence electronically and the use of UML for the development of safety-critical systems.

The need for constructing certification evidence electronically has been identified by [5] and [18] in order to manage the complexity and large amounts of information that needs to be collated. Lewis [18] calls for an underlying information model to manage the complex links that exist between the various pieces of safety evidence. We propose an approach that takes into account both the above points. Our conceptual model provides the underlying information model for a standard and our profile provides a practical mechanism for using this information in order to create the relevant artifacts for certification. Cockram and Lockwood [5] present a proprietary tool that uses hypertext for linking all the diverse pieces of information for certification. This enables them to link information but the links are manually created without an underlying information model to guide the creation of the evidence as we do. We use model-driven engineering technologies, and in particular UML profiles, to aid with the creation of this evidence for a particular standard. The profile can be exported and used in any UML modelling tool and the constraints provide guidance when elaborating the domain model. The set of inter-related information items can be seen via the elaborated models and the instance models use uniform resource locators to link to artifacts directly.

The use of model-based technologies, especially UML, is gaining pace in the development of safety-critical software. The Object Management Group (OMG) has standardized the UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [26] and the UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms (QFTP) [25]. Both these profiles are used for modelling the real-time and performance properties of safety-critical systems. Similarly, Berkenkotter [2] and Hannemann [3] have created a profile for the railway domain that aids the design and verification of interlocking functionality. However, neither of these are meant to characterize the evidence requirements of a standard according to which safety-critical systems are certified.

A profile that deals with certain aspects of certification is proposed by Zoughbi et. al. [37]. Their profile enables the direct addition of certification information to software models for compliance with the RTCA DO-178B standard [31], used in commercial and military aerospace software. However, this profile is targeted at maintaining traceability between requirements, design and code, which is only a part of the recommendations of DO-178B. The profile that we propose deals with a complete standard and takes into account not only evidence regarding requirements and design but also the wide range of concepts related to the management of the development process in safety-critical systems.

Huhn and Hungar [10] discuss the proliferation of UML in the model-based development of safety-critical software. 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 has not been adequately addressed. Our profile is a starting point for addressing this gap.
Recently, the OMG has put forward a proposal, called the Software Assurance Evidence Metamodel (SAEM) [27], for managing safety assurance evidence. 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 approach 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 as we have proposed). On the other hand, a profile of the SAEM could be incorporated into our approach and cover both the evidence requirements for compliance as well as the evaluation of the evidence to ensure that it is sufficient to substantiate the safety claims. Together, these could be means to further the field of model-based certification.

Regarding compliance to a specific standard, Chung et. al.[4] study the problem of compliance of a user-defined workflow with the activities prescribed in IEC61508. They check (process) compliance by comparing user-defined activities in an organization against models of the activities in the standard. This work is similar to what we propose in that the aim is to model compliance information; however, we go beyond the process aspects of a standard and provide an evidence information model for the entire standard which in turn is the basis of our profile that is used to manage certification evidence.

Finally, we have used UML profiles of safety related standards in prior work [30], 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 [30] as a way to keep track of the relationships between two standards – a generic and a sector-specific one.

7. Conclusion and Future Work

In this article, we described an approach based on model-driven engineering principles and technology to specify and analyze the safety evidence required for compliance to safety standards. 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 so by capturing the relevant standard as a conceptual model in the UML notation and using the resulting model as a basis for creating a UML profile. The profile is augmented with constraints expressed in the OCL language 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. The constraints enforced by the profile can be automatically checked by existing OCL constraint engines.

Developing conceptual models along with glossaries of definitions for standards help avoid the ambiguity issues that can exist in text-based standards. The elaboration phase in our proposed approach provides step-by-step guidance on how to align the concepts in the system to the relevant standard and create the relevant evidence items. The domain models allow different levels of abstraction to be expressed. Thus, we are able to provide an overall picture of the system and at the same time the flexibility
to drill down to a specific part of the system as necessary, while being aware of the connections between the different pieces. The different levels of abstraction and the breakdown of the system via the domain models also proves useful in collaborative work. Each engineer can work on the part of the system assigned to them, and create and elaborate the models as necessary. The changes are visible to the other engineers but need not interfere with their work unless there is some overlap.

We have applied our approach in a pilot study in the context of sub-sea production control systems. The case study shows that our approach is feasible in a realistic environment and that it can provide useful guidance in relating the concepts in the domain of sub-sea production control systems to those of the IEC61508 standard. The case study further shows that the effort involved in applying our approach was acceptable. Since IEC61508 is a generic standard that applies to multiple domains, a successful application of our approach to it is a good indication of the usefulness and wider relevance of our work.

We have further conducted a survey of industry practitioners. The results show that half of the domain experts we surveyed found IEC61508 difficult to understand and only 30% found it usually easy to use for certification. When presented with a conceptual model of the standard, 92% of the participants found the conceptual model to be either very easy, easy, or averagely easy to understand while only 50% found the textual standard averagely easy to understand. When surveyed about our approach, no one found it difficult to follow the steps of the approach and 92% of participants were in favour of adopting the approach in their work.

In future work, we would like to add a report generator to present the models in the form of reports for upper management. This would allow the provision of relevant information for a number of different actors in an organization. The approach itself can be extended for the certification of a system to multiple inter-related standards. We would also like to study whether conceptual models can be used as means for presenting standards and whether the use of models would make it easier to identify and take action on changes in the standards when they are updated. Finally, we would like to extend our work to help the certification body with the evaluation of evidence collated according to our approach.

Acknowledgments. We are grateful to the staff at the partner company where we conducted our case study and survey for volunteering their time and sharing their knowledge. Funding for this research was provided by DNV under the ModelME! project, by the National Research Fund - Luxembourg (FNR/P10/03 - Validation and Verification Laboratory), by the FP7 Programme under Grant Agreement No. 289011 (OPENCOSS), and by the Research Council of Norway under Project No. 203461/030 (Certus).

References


Appendix A. The IEC61508 Conceptual Model

The conceptual model for the IEC61508 is shown in Figure A.30 as a UML class diagram. The conceptual model has a total of ten packages, containing abstractions for modelling the main concepts of IEC61508. We briefly explain each package. For more details, see [28]. The **System Concepts** package describes 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).

Along with the conceptual model, a glossary was created for each concept and relationship, a part of this glossary, describing the most important concepts is shown in Table A.7.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>A unit of behaviour in a process.</td>
</tr>
<tr>
<td>Agent</td>
<td>A person or organization that has the capability and responsibility for carrying out an activity.</td>
</tr>
<tr>
<td>Artifact</td>
<td>One of the many kinds of tangible by-products produced during the development of a system.</td>
</tr>
<tr>
<td>Assumption</td>
<td>A premise that is not under the control of the system of interest, and is accepted as true without a thorough examination. Assumptions can, among other things, be related to the environment of the system, the users, and external regulations.</td>
</tr>
<tr>
<td>Block</td>
<td>Entity of hardware or software, or both, capable of accomplishing a specified purpose.</td>
</tr>
<tr>
<td>Change</td>
<td>A modification made to the PES, Block or Artifact.</td>
</tr>
<tr>
<td>Competence</td>
<td>The ability to perform a specific task, action or function successfully.</td>
</tr>
<tr>
<td>ControlledItem</td>
<td>A PES, Block or Artifact for which meaningful increments of change are documented and recorded.</td>
</tr>
<tr>
<td>Defect</td>
<td>An error, failure, or fault in a system that produces an incorrect or unexpected result, or causes it to behave in unintended ways.</td>
</tr>
<tr>
<td>Description</td>
<td>A planned or actual function, design, performance or activity (e.g., function description).</td>
</tr>
<tr>
<td>DesignatedState</td>
<td>The state of the EUC related to safety, the EUC is either in a safe state or an unsafe state.</td>
</tr>
<tr>
<td>Diagram</td>
<td>Specification of a function by means of a diagram (symbols and lines).</td>
</tr>
<tr>
<td>Enhancement</td>
<td>Provision of improved, advanced, or sophisticated features.</td>
</tr>
<tr>
<td>Error</td>
<td>Discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition.</td>
</tr>
</tbody>
</table>
Figure A.30: The IEC61508 Conceptual Model
<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>A single occurrence in a series of occurrences that cause a hazard to occur.</td>
</tr>
<tr>
<td>Failure</td>
<td>Termination of the ability of a functional unit to perform a required function.</td>
</tr>
<tr>
<td>Fault</td>
<td>Abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function.</td>
</tr>
<tr>
<td>GeneralStandard</td>
<td>A standard that provides generic recommendations on a specific subject to a number of related domains.</td>
</tr>
<tr>
<td>HazardBlock</td>
<td>Any entity of hardware – this may be mechanical, electrical or electronic that is used in the composition of the system.</td>
</tr>
<tr>
<td>HazardousElement</td>
<td>The basic hazardous resource creating the impetus for the hazard, such as a hazardous energy source such as explosives being used in the system.</td>
</tr>
<tr>
<td>Hazard</td>
<td>Any real or potential condition that can cause injury, illness, or death to personnel damage to or loss of a system, equipment or property or damage to the environment.</td>
</tr>
<tr>
<td>Individual</td>
<td>Refers to a person.</td>
</tr>
<tr>
<td>InitiatingMechanism</td>
<td>The trigger or initiator event(s) causing the hazard to occur. The IM causes actualization or transformation of the hazard from a dormant state to an active mishap state.</td>
</tr>
<tr>
<td>Instruction</td>
<td>Specifies in detail the instructions as to when and how to perform certain jobs (for example operator instruction).</td>
</tr>
<tr>
<td>Interface</td>
<td>An abstraction that a block provides of itself to the outside. This separates the methods of external communication from internal operation.</td>
</tr>
<tr>
<td>Issue</td>
<td>A unit of work to accomplish an improvement in a system.</td>
</tr>
<tr>
<td>List</td>
<td>Information in a list form (e.g., code list, signal list).</td>
</tr>
<tr>
<td>Log</td>
<td>Information on events in a chronological log form.</td>
</tr>
<tr>
<td>Mistake</td>
<td>Human action or inaction that can produce an unintended result.</td>
</tr>
<tr>
<td>HardwareBlock</td>
<td>The different modes that a system can be operating in, e.g. normal, maintenance, test, emergency.</td>
</tr>
<tr>
<td>OperatingMode</td>
<td>A set of activities with determined inputs and output that are carried out at a specific time during the life of a system.</td>
</tr>
<tr>
<td>Organization</td>
<td>A social arrangement which pursues collective goals, which controls its own performance, and which has a boundary separating it from its environment.</td>
</tr>
<tr>
<td>Phase</td>
<td>System for control, protection or monitoring based on one or more programmable electronic devices, including all elements of the system such as power supplies, sensors and other input devices, data highways and other communication paths, and actuators and other output devices.</td>
</tr>
<tr>
<td>Plan</td>
<td>System for control, protection or monitoring based on one or more programmable electronic devices, including all elements of the system such as power supplies, sensors and other input devices, data highways and other communication paths, and actuators and other output devices.</td>
</tr>
<tr>
<td>Plan</td>
<td>Explanation of when, how and by whom specific activities shall be performed (e.g., maintenance plan).</td>
</tr>
<tr>
<td>Programmatic-</td>
<td>Any physical entity based on computer technology which may be comprised of hardware, software, and of input and/or output units.</td>
</tr>
<tr>
<td>ElectronicSystem</td>
<td>The state of the EUC when safety is achieved.</td>
</tr>
<tr>
<td>Programmatic-</td>
<td>Any physical entity based on computer technology which may be comprised of hardware, software, and of input and/or output units.</td>
</tr>
<tr>
<td>HardwareBlock</td>
<td>The fundamental reason or reasons serving to account for something.</td>
</tr>
<tr>
<td>Rationale</td>
<td>Sound practices and guidance for the achievement of a particular objective.</td>
</tr>
<tr>
<td>RecommendedPractices</td>
<td>The results of activities such as investigations, assessments, tests etc. (e.g., test report).</td>
</tr>
<tr>
<td>Request</td>
<td>A description of requested actions that have to be approved and further specified (e.g., maintenance request).</td>
</tr>
<tr>
<td>Requirement</td>
<td>A necessary attribute in a system; a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a user.</td>
</tr>
<tr>
<td>ResidualRisk</td>
<td>Risk remaining after protective measures have been taken.</td>
</tr>
<tr>
<td>Risk</td>
<td>Combination of the probability of occurrence of harm and the severity of that harm.</td>
</tr>
<tr>
<td>SafeState</td>
<td>The state of the EUC when safety is achieved.</td>
</tr>
<tr>
<td>SafetyIntegrity-Level</td>
<td>The probability of a safety-related system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time.</td>
</tr>
<tr>
<td>SafetyRequirement</td>
<td>A prescriptive statement that ensures that the system carries out its functions in an acceptably safe manner.</td>
</tr>
<tr>
<td>SectorSpecific-</td>
<td>A standard that provides recommendations for a specific industrial sector (e.g., the energy sector).</td>
</tr>
<tr>
<td>HardwareBlock</td>
<td>Any entity of software that may be used for controlling the system – this may be embedded or application software or even different levels of software such as module, component, subsystem, system.</td>
</tr>
</tbody>
</table>

Continued from previous page ...
Appendix A.1. Software Lifecycle Activity Packages

The IEC 61508 standard prescribes certain safety lifecycle activities for software. In this section, we show how we have modelled the activities recommended for the software lifecycle along with the input and output artifacts of each activity.

Software Safety Requirements Definition. The goal of this activity to specify the requirements for software safety in terms of the requirements for software safety functions and software safety integrity. See Figure A.31.

Software Safety Validation Planning. The goal of this activity is to develop a concrete plan for validating the software in terms of safety. See Figure A.32.

Software Architecture Development. The goal of this activity is to create a software architecture that fulfills the specified requirements for software safety with respect to the required safety integrity level. See Figure A.33.

Support Tools and Coding Standard Development. The goal of this activity is to select a suitable set of tools, including languages and compilers, for the required safety integrity level over the whole safety lifecycle of the software. See Figure A.34.

Software System Design Development. The goal of this activity is to design software that fulfills the specified requirements for software safety with respect to the required safety integrity level, which is analyzable and verifiable, and which is capable of being safely modified. See Figure A.35.
Figure A.32: Software Safety Validation Planning

Figure A.33: Software Architecture Development

Figure A.34: Support Tools and Coding Standard Development
Software Module Design Development. The goal of this activity is to design the software modules that fulfill the specified requirements for software safety. See Figure A.36.

Source Code Implementation. The goal of this activity is to implement software that fulfills the specified requirements for software safety. See Figure A.37.

Software Module Testing. The goal of this activity is to verify that each software module performs its intended function and does not perform unintended functions. See Figure A.38.

Software Integration Testing. The goal of this activity is to verify that all software modules, components and subsystems interact correctly to perform their intended functions and do not perform unintended functions. See Figure A.39.

Programmable Electronics Integration. The goal of this activity is the integration of the software onto the target programmable electronic hardware and to combine the
Figure A.37: Source Code Implementation

Figure A.38: Software Module Testing

Figure A.39: Software Integration Testing
software and hardware in the safety-related programmable electronics system to ensure
their compatibility and to meet the requirements of the intended safety integrity level.
See Figure A.40.

Software Operation Procedures Development. The goal of this activity is to provide
information and procedures necessary to ensure that the functional safety is maintained
during the operation of the system. See Figure A.41.

Software Modification Procedures Development. The goal of this activity is to provide
information and procedures necessary to ensure that the functional safety of the system
is maintained during modification of the software. See Figure A.42.

Software Safety Validation. The goal of this activity is to ensure that the integrated
system complies with the specified requirements for software safety at the intended
safety integrity level. See Figure A.43.
Figure A.42: Software Modification Procedures Development

Figure A.43: Software Safety Validation

Pg 33 on 61508-3 9.5 all kinds of artifacts can be an input to this activity.
Cross-Cutting Activities: Software Modification, Verification, and Functional Safety Assessment. These three activities (shown in Figure A.44) link to all other activities being performed and can thus potentially affect all of them:

- **Software Modification**: The purpose of this activity is to ensure that corrections, enhancements or adaptations to the validated software sustain the required software safety integrity level and follow the modification procedures.

- **Software Verification**: The activity is used, to the extent required by the safety integrity level, to test and evaluate the outputs from a given software safety life-cycle activity to ensure correctness and consistency with respect the standards and the provided inputs.

- **Software Functional Safety Assessment**: The purpose of this activity is to investigate and arrive at a judgment on the functional safety achieved by the system.

Appendix B. Domain Model of a Sub-Sea Production System

In Figure B.45, we present a high-level domain model of a sub-sea production system. This was the model created during the case study described in Section 5.1.5
Figure B.45: A High-level Domain Model of a Sub-sea Production System