An Ontology for Safety Assurance Information Management

Camilo Almendra
Departamento de Computação,
Universidade Federal do Ceará
Fortaleza, Brazil
camilo.almendra@ufc.br

Carla Silva
Centro de Informática,
Universidade Federal de Pernambuco
Recife, Brazil
ctlls@cin.ufpe.br

David Campelo
Robert Bosch GmbH
Ovar, Portugal
david.campelo@bosch.com

ABSTRACT

Context: Diverse information needs to be traced and managed throughout the development lifecycle of safety-critical systems, such as requirements, design definitions, tests and code. A safety-critical system (SCS) undergoes certification procedures that require the presentation of arguments supporting the design and implementation decisions involved in the SCS construction. Nowadays, many regulations require elaborating assurance cases (ACs) to present such arguments. The integrated management of the diverse data required to produce the ACs can leverage the continuous traceability assessment and support the automated generation of ACs at any time in the SCS lifecycle. **Objective:** Ontologies can integrate metamodels to connect data from different domains. We designed and implemented a Safety Assurance Ontology (SAO) to integrate outputs of safety and software engineering activities. SAO encompasses an information model, rules and competency questions to support the work of software, safety and assurance engineers in the context of an SCS development that requires the production of ACs. **Method:** SAO was built based on knowledge extracted from safety datasets and argumentation patterns. This work presents the SAO's design, implementation, and evaluation through an illustrative scenario with public datasets and an expert survey with industrial practitioners and researchers. **Results:** SAO comprises concepts, relationships, and formation rules integrating safety assurance and software project information. We implemented the ontology information model and its formation rules using semantic web technologies. **Conclusions:** SAO can trace relevant project information, infer implicit information, and highlight traceability gaps in the public datasets evaluated. Practitioners find SAO's competency questions on traceability gaps useful in revealing direct and indirect relationships among project items. They also find that the basic competency questions facilitate the contextualisation of items traced in the project repositories. Also, they pointed out that adding specialisations for key concepts can enhance practitioners' grasp of the metamodel scope, thus promoting its applicability.

KEYWORDS

Assurance cases, Ontology, Safety-critical systems

1 Introduction

Safety-critical systems (SCS) are developed under guidance of and constrained by regulations (e.g. norms or standards). In the certification procedure, product manufacturers need to demonstrate that an SCS is fit for use by showing how they addressed the regulatory requirements. Safety assurance case (AC) is an approach being adopted by regulations to communicate and argue over the critical properties of a system [31]. The main purpose of an AC is to provide a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment [30]. Nowadays, some regulations mandate or recommend elaborating an AC as part of the certification documentation:

- Defence Standard 00-056 – Safety Management Requirements for Defence Systems – Part 2 [36];
- Cenelec EN 50129 – Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling [10];
- ISO 26262 – Road vehicles – Functional safety – Part 10 [28];
- Infusion Pumps Total Product Life Cycle – Guidance for Industry and FDA Staff [20]; and
- ISO TS 81001-2-1 Health software and health IT systems safety, effectiveness and security – Part 2-1 – Guidance for the use of assurance cases for safety and security (under development) [29].

AC encompasses the tracing of links among requirements, design, code, tests, hazards, risks, cause of hazards, and other artefacts [35]. Thus, requirements traceability management is a strategic process to ensure that software is safe for use and to comply with AC development needs. However, maintaining the traceability among safety requirements and other artefacts alongside the system development lifecycle is still challenging [47].

An essential information that needs to be included in ACs is the assurance rationale [3, 27, 42]. Assurance rationale is the reasoning underlying safety-related project decisions, which is essential to complement the tracing information. Registering the assurance rationale and relating it to the project information comprise a recurring difficult faced in AC development, that may be addressed by the use of information models [15]. The verification of consistency of the project information with regard to assurance rationale, and how to structure the arguments in the AC are also recurring concerns, and are the focus of this work.

We developed an approach to assess the consistency and to classify the assurance rationale and project information, to support developers on the production of the ACs. Our approach relies on an Safety Assurance Ontology (SAO) to perform consistency checking, and to derive implicit classification and relationships. SAO could provide pre-assessment of the traceability information managed inside project repositories, without the need to produce the ACs. Also, the automated classification infers hidden relations between project items, thus improving the information available to produce the ACs.

Our ontology is based on knowledge extracted from AC datasets and patterns, and on an Assurance Rationale Model [3]. The contributions of this work are:

- The identification of formation rules for assurance rationale and project information extracted from AC datasets and patterns; and
- An operational ontology named Safety Assurance Ontology (SAO) that implements an assurance information model and the rules of formation, using description logic and its associated technologies (open artefacts).
- An evaluation of the ontology through an illustrative scenario with public datasets, and an expert survey evaluation.

This paper is organised as follows. In Section 2, we discuss background concepts and related works. In Section 3, we present the ontology development. In Section 4, we discuss the evaluation of the ontology. Finally, we discuss our conclusions and state future works in Section 5.

2 Background and Related Works

2.1 Assurance Evidence Management

What should compose an Assurance Case (AC) and how it should be structured varies depending on the target standards and AC notations used. This variation poses difficulties in reusing, within the same company, the know-how and assets across products targeting different domains or multiple domains at once [3, 27, 41].

We now illustrate example of assurance information items in the context of a safety-critical project using the Open PCA Pump dataset [33, 34]. The dataset contains a requirements specification [34] and an assurance case report [33]. The following excerpt illustrates key assurance information that is commonly only recorded in the AC artefacts.

Figure 1 illustrates project items that are commonly used to elaborated safety arguments in the AC artefacts. The items "*Rationale 2.2.A.5*" and "*Strategy 2.2.A.5*" explain the argument behind why the single requirement "*Patient bolus will not be delivered until minimum time between boluses has expired*" is sufficient to mitigate the hazard "Too many user boluses". The requirements specification [34] provided in the dataset does not state any hazard explicitly, and only in the AC report [33] we can find the relationship between requirements and hazard described.

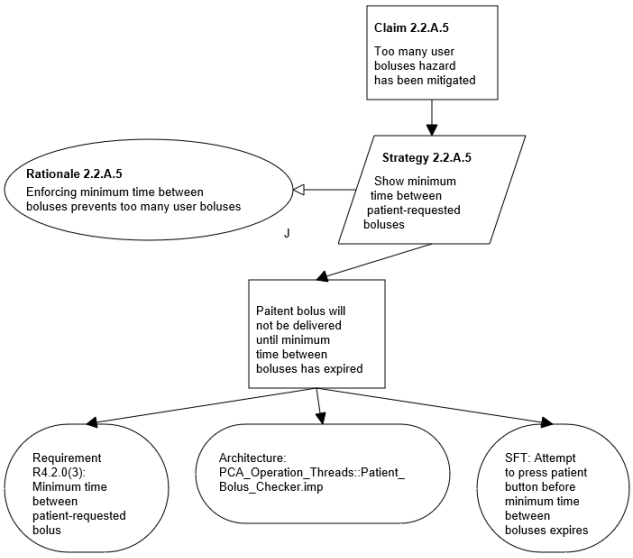

Figure 1: Argument example (adapted from [33])

2.2 Requirements Traceability

Certification of a system requires evidence and information from all development disciplines. AC development requires that diverse pieces of information be logically linked together to elaborate compelling arguments. As Cleland-Huang et al. [11] found, practitioners who do not work in regulated domains tend to be unfamiliar or non-believers of software traceability traits. This is another discipline to be promoted, as it is essential to support a cost-effective AC development.

Safety is a non-functional requirement and a cross-cutting concern directly related to system components and behaviour. To meet safety assurance goals, traceability must be supported. Otherwise, several problems related to traceability could delay the certification of Safety-Critical System (SCS) products, such as lack of a traceability information model, uneven granularity of tracing links, and missing or redundant links [35]. The information needed to build a compelling AC can be compared to a high-end usage of requirements traceability [40]. SCS demands the application of traceability for full lifecycle coverage, capture of decisions, rationale, and tracing across product and process dimensions, and requirements decomposition [40].

Exploring and registering rationale for non-functional requirements often involves modelling alternatives to assist comparative analysis [9]. Current AC templates and specification languages provide support only to express the chosen requirements and design solutions, as well as the arguments for their positive contribution to safety. However, a compelling argumentation favouring a selected design is often backed by demonstrating its advantages compared with other alternatives. The management of rationale and the mechanisms to translate it into arguments that are part of an AC are key activities.

Assurance cases naturally replicate information recorded elsewhere in the project repositories or artefacts. Such duplication of information may increase cost and inconsistencies as the project scales. It is essential to provide means for a partially or fully automated composition of cases from application lifecycle management systems. Such management systems may need to be enhanced with safety-related information models [14]. These information models may support a full model-based approach in which a single tool concentrates all data (e.g., OpenCert [19]), or more lightweight approaches that retrieve project data and perform specific computations.

2.3 Ontology engineering

Project management tools need to be perceived as knowledge repositories, especially with regard to requirements traceability information. Ontology-based approaches have been explored in the context of Requirements Engineering (RE) for the following [46]:

- Support the structuring and recovery of knowledge;

- Verification and validation;
- Support to understand/identify concepts;
- Control/share of vocabulary; and
- Integration and transformation models.

Ontology is an explicit specification of a shared conceptualisation that holds in a particular context [44]. It aims to organise domain knowledge over individuals, objects, types, attributes, relationships and functions. Once organised in a knowledge base, this information can be easily shared among people involved in the context of the ontology, such as users, customers, requirements analysts, architects and any other personnel involved in Software Engineering.

In our ontology-based approach, we aim to the following benefits:

- Make domain premises explicit: safety and software teams are implicitly considering domain premises while carrying out their activities, so an ontology can foster the exchange of knowledge among them.
- Analyse domain knowledge to compute implicit knowledge by invoking reasoning: the interrelation between safety assurance and regular project items may not be trivial or easy to handle adequately on a large scale, so the use of an ontology can support the identification and gathering of data to develop the ACs.
- Enable reuse of domain knowledge: SCS systems tend to be developed on top of other systems and components, so the ontology can be used to interrelate information extracted from different project repositories.

Ontologies may be implemented and operationalised in many ways. We used Description Logic (DL) as our logic system as it is sufficiently expressive to represent project information, and the core reasoning for DLs is mostly decidable (i.e. reasoner will not run forever) [49]. The use of an ontology implemented on top of Ontology Web Language (OWL) technology benefits from three main capabilities: consistency checking, automatic classification, and ease of extension/tailoring.

One way to input data (as individuals) into ontology systems is through set of triples. A semantic triple (or RDF triple) is the atomic data entity that enables any knowledge to be represented in a machine-readable way [49]. An important feature of RDF is that almost any kind of knowledge repository may be represented as a collection of RDF triples (RDF graph or just RDF) [49]. For example, a collection of issues from an issue tracking system may be represented as an RDF graph (Figure 2). The predicates used shall comply with a defined schema (semantic); in the example, the predicates “type”, “label”, and “description” are based on the RDF Schema [8], while the “mitigates” refers to a relationship type in the SAO ontology.

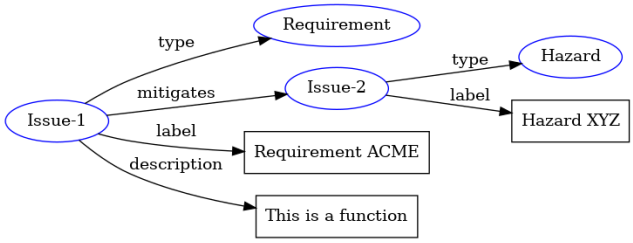

Figure 2: RDF graph example

2.4 Related works

Denney and Pai [16] proposed a lightweight method for automatically generating fragments of an AC based on information extracted from an existing set of artefacts. The approach retrieves data from multiple tables, which can be considered an information model. However, these tables do not comprise concepts related to requirements, design or assurance rationale. Also, it does not support neither consistency checking nor classification of implicit knowledge. Predefined assumptions and strategies are placed in the generated argument fragments.

Armengaud [5] proposed a solution for AC generation based on the extraction of information coming from existing work products. The approach comprises a wrapping step when data from various sources in the development process are extracted to be ingested in a database. Wrappers can be implemented for the extraction. The second step is the harmonisation of data, that is, when the data from various sources are semantically interconnected and mapped to the metamodel defined by the approach. Finally, the data harmonised is used to generate some reports, including an AC report. The purpose is to monitor development status and perform some checks, such as completeness of description (are the elements complete – e.g., all requirement fields filled) or completeness of traces (are all the traces available). There is no support to register relationships from hazards and requirements to assurance information, such as assumptions, justifications, and strategies. There is no support for discovering implicit knowledge.

Cleland-Huang and Vierhauser [12] and Agrawal et al. [1] proposed the SAFA approach, which generates ACs from project traceability data, allows assurance-related information to be included in the generated ACs, and finally, supports the management of these ACs upon the evolution of the project data. The approach is based on a traceability information model that is implemented inside an issue tracker. The information model covers safety-related concepts such as hazards and links; but it does not cover justification and strategies (only assumptions), which are key to assemble final assurance arguments. Consistency checking is available, although there is no automatic classification of implicit knowledge.

Existing approaches partially automate AC generation but either do not i) include consistency checking, ii) generate automatic classification, or iii) provide traceability questions for systematic gap discovery. SAO covers all three aspects. These approaches manage most of the assurance information outside the information model, thus directly inside the ACs, which are external artefacts maintained alongside the development. A consequence is that the data required to elaborate (manually or automatically) an AC is not entirely available in the main information model of the discussed approaches, demanding that practitioners manually synchronize and maintain AC documentation.

In our work, we aim to design an unifying information model that integrates project and assurance information, and to provide consistency, inference and completeness checks.

3 Safety Assurance Ontology

In previous work [2, 4], we proposed the use of an external ontology-based information system to extract and assess project traceability, to help teams detect and solve deviations in their traceability data continuously throughout the Software Development Life Cycle (SDLC). In this work, we present, discuss and make available our ontology, as it can be embedded in other solutions or used through an ontology editor.

SAO is designed to be used in a context where team members seek to leverage the integrated assessment of regular and assurance-related project information from diverse tools. Tools such as issue trackers are very good to register requirements, tests, design decision and many other types of project information, and also to register the traceability among them. However, the identification of gaps and inconsistencies that could hinder the safety of the system under development is not provided by such tools.

According to Figure 3, SAO can support automatic reasoning to check for inconsistencies and discover implicit information using inference from the information recorded in the project management tools. Then, team members can adjust gaps and/or inconsistencies that could hinder the quality of assurance cases and, therefore, jeopardize certification procedures.

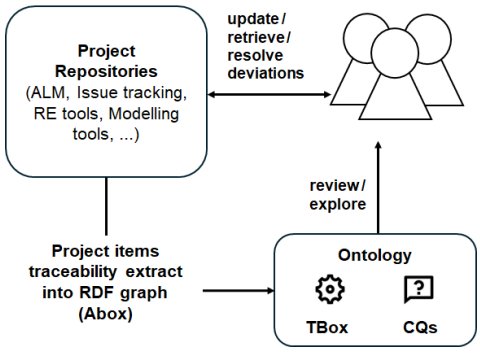

Figure 3: Usage context of the Safety Assurance Ontology

We state some design goals for the ontology. First, we aim to support the integrated management of assurance and regular project information. So, our ontology should handle these two domains in the same metamodel. Second, we aim to support the automated assessment of consistency and completeness. Finally, we aim to allow easy customisation of the ontology for its adoption in different SCS project settings.

We decided to address knowledge management of assurance and project information using a system of logic that supports the Open-World Assumption (OWA). OWA is useful for scenarios in which knowledge is managed as it is discovered, that is, the complete information is not available at once. This is the scenario in the early stages of a software project system development when much information needed for the final certification procedures is yet to be produced.

We chose to use description logic (DL) as our logic system. It is sufficiently expressive to represent project data repositories, and the core reasoning for DLs is mostly decidable (i.e. reasoner will not run forever) [49]. As efficient reasoner tools are available [49], we can rely on this technology to build predictable tools for general-purpose uses. OWL is one of the most common languages used for ontologies related to software engineering [17] and is compatible with DL.

We designed the SAO, an operational ontology implemented in OWL that models the concepts, relationships and constraints of the information model. Some consistency rules are implemented directly as axioms, while other consistency and completeness rules are implemented using SPARQL, a semantic query language that can be applied in RDF.

The base version of the SAO provides a lean (yet precise) set of definitions and rules that address core aspects of safety-critical software projects. To have practical applicability, this technology needs to be extensible to address the idiosyncrasies of a company development process. Extensibility is inherent to semantic web technologies; thus, our ontology base version can readily adapt to a company's needs (or even use different versions across projects).

An ontology based in semantic web technologies can be used in editor tools like Protégé or be embedded into an information system using the many third-party libraries available [4, 39].

The methodological steps below were based on discussions of ontology building reported by Gómez-Pérez and Benjamins [24], knowledge engineering principles defined by Schreiber [44] and the METHONTOLOGY process [21]. The steps are the following:

- Requirements specification: Define the scope of ontology and its intended use;
- Knowledge gathering: Conduct a literature review to identify knowledge sources;
- Implementation: Conduct iterative cycles of conceptualisation, integration, implementation and verification of ontology prototypes; and
- Evaluation: Demonstrate ontology usage through an illustrative scenario and experts opinion survey.

It is important to point out that this is not a linear, cascading process. Each step requires a review of definitions made in previous steps, so it is an iterative and incremental process.

3.1 Specification

Purpose and Scope. The purpose of an ontology is to integrate common and interrelated knowledge of two systems engineering practices: safety engineering and software engineering. We refer to the conceptual model of our ontology as "information model", which is developed alongside the ontology specification. The intended use of this ontology is to serve as a base system to integrate outputs of safety and software engineering activities performed by all kinds of personnel involved in safety-critical system development (e.g., developers, safety/assurance engineers). To support users' needs, we formulated competency questions that the framework should answer concerning the project information provided.

Intended Users and Scenarios of Use. The intended use of this ontology is to serve as a base system to integrate outputs of safety and software engineering activities performed by all kinds of personnel

involved in SCS development (e.g., developers, safety engineers, assurance engineers). Intended users and their goals are the following:

- *Software engineers* working on requirements, design, tests and other kinds of artefacts who need to assess the consistency of software specifications and completeness of traceability with safety assurance artefacts.
- *Safety engineers* working on safety analysis who need to assess the consistency of safety specifications and completeness of traceability with software specifications.
- *Assurance engineers* producing ACs or other certification documents who need to retrieve project information and assess its correctness and completeness with regard to safety assurance argumentation.

The scenarios of use are all related to actions users can perform in the knowledge repository: i) register project information items (e.g., requirements, design definitions), ii) register safety analysis and assurance argument information items (e.g., hazards, environmental contexts, assumptions), iii) register traces between items, iv) review provided (explicit) items and their relationships, v) review inferred (implicit) classifications and relationships for provided items, vi) and assess correctness and completeness.

This ontology is classified as a *Task-Specific Ontology* because it embodies just the conceptualisations that are needed for carrying out a particular task [44].

Sources of Knowledge. The knowledge gathering comprises a literature review to identify knowledge sources, such as foundational ontologies, related domain-specific ontologies, international standards, datasets, and other metamodels. We chose two metamodels [40, 48] and a standard [25] that serve as domain-specific ontologies for connecting our ontology (depicted in Figure 4) to both safety assurance and software traceability domains. As basic vocabulary, we used the RDS Schema, a data-modelling vocabulary for RDF data [8]. We also used AC datasets as sources. These sources are presented in Table 1. An extended discussion of conceptualisation of this information model is present in [3].

Table 1: Knowledge sources

Source	Main contributions
Traceability reference model [40]	Core traceability concepts
Safety requirements metamodel [48]	Core safety concepts applied to software systems
GSN standard [25]	AC concepts and rules of formation
RDF Schema [8]	Basic vocabulary to represent objects, object attributes and links in RDF
OpenPCA Pump project [26]	Dataset with ACs and project information
GPCA Pump project [22]	Dataset with ACs and project information
GIIP project [22]	Dataset with project information

Competency Questions. A competency question is a natural language question that the ontology should be able to answer. Questions and their responses serve as a requirements specification against which the ontology can be evaluated [45]. Thus, the questions were specified towards the informational needs of team members reviewing assurance information or constructing ACs. Also, the questions encompassed traceability gaps that impact the quality of the arguments. They were identified by analysing each kind of relationship that each concept may or should have in a consistent and complete model instance. Therefore, these questions guided the implementation of the internal data model and the data queries.

Table 2 lists all the competency questions that comprise the specification of SAO. From the competency questions, we devised types of items, relationships and attributes that need to be handled. Additionally, we extracted rules and constraints from the metamodels on which our information model is based. We organise the questions in two types: basic and traceability gaps.

Table 2: Competency questions

	Basic questions ("Which are the...")
CQ01.	... Objects and their classifications?
CQ02.	... Relationships associated with an Object?
CQ03.	... Hazards?
CQ04.	... Safety Requirements?
CQ05.	... Requirements?
CQ06.	... Design Definitions?
CQ07.	... Causes?
CQ08.	... Contexts?
CQ09.	... Components?
CQ10.	... Assumptions?
CQ11.	... Justifications?
CQ12.	... Strategies?
CQ13.	... Sources?
	Traceability gaps questions("Which are the...")
CQ14.	... Requirements that are not allocated in any Component?
CQ15.	... Safety Requirements that do not mitigate any Hazard?
CQ16.	... Safety Requirements that are not explained by any Rationale?
CQ17.	... Hazards that are not mitigated by any Safety Requirement?
CQ18.	... Causes that are not cause of any Hazard?
CQ19.	... Hazards that have no Cause associated?
CQ20.	... Hazards that are not explained by any Rationale?
CQ21.	... Hazards that are not contextualized by any Context?
CQ22.	... Rationales that do not explain any Object?
CQ23.	... Rationales that are not classified as any of the subclasses?
CQ24.	... Contexts that do not contextualize any Object?
CQ25.	... Components that are not allocated to any Requirement?

Basic questions are focus on retrieving project items by type (e.g. "CQ04: Which are the Safety Requirements?") or extracting all adjacent items given a specific item (e.g. "CQ02: Which are the relationships associated with an Object?"). Traceability gaps are focus on finding project items for which some key relationships are missing (e.g. "CQ17: Which are the Hazards that are not mitigated by any Safety Requirement?").

Information Model. The ontology handles the project information at the granularity of single, uniquely identified items. The ontology also handles the traceability between the project items. It is out of the scope of this work to discover or handle knowledge inside the natural language texts of the project items. Each concept and relationships is fully described in the file `sao.owl` (see Artefact Availability). Figure 4 presents the information model.

Besides the definition of concepts and relationships, the ontology specifies the rules listed in Table 3. Rules from 1 to 20 are not restrictions, but instead, classifications that will be inferred if the ontology reasoner finds items/links matching these conditions. Finally, rules 21 and 22 are restrictions, and if an object is classified as more than one of the disjointed classes, then an inconsistency should be triggered by the framework.

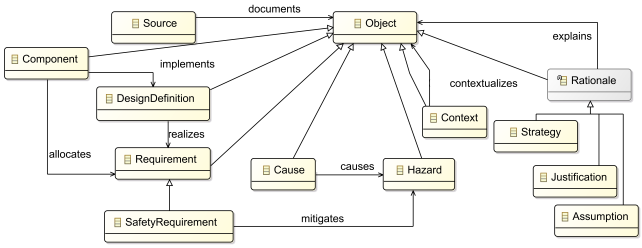

Figure 4: Information metamodel

Table 3: Formation and classification rules

Rule	Description
1.	**Object** that *allocates* other is a **Component**
2.	**Object** that *is allocated in* other is a **Requirement**
3.	If A *refines* B and B *is allocated in* C, then A *is allocated in* C
4.	**Object** that *causes* other is a **Cause**
5.	**Object** that *is caused by* other is a **Hazard**
6.	If A *contextualizes* B and B *is refined by* C, then A *contextualizes* C
7.	**Object** that *contextualizes* other is a **Context**
8.	**Object** that *documents* other is a **Source**
9.	**Object** that *explains* other is a **Rationale**
10.	**Object** that *implements* other is a **Component**
11.	**Object** that *is implemented by* other is a **Design Definition**
12.	**Object** that *documents* other is a **Source**
13.	**Object** that *mitigates* other is a **Safety Requirement**
14.	**Object** that *is mitigated by* other is a **Cause** or **Hazard**
15.	If A *mitigates* B and B *causes* C, then A *mitigates* C
16.	If A *refines* B and B *mitigates* C, then A *mitigates* C
17.	**Object** that *realizes* other is a **Design Definition**
18.	**Object** that *is realized by* other is a **Requirement**
19.	**Object** that *refines* or *is refined by* a **Hazard** is a **Hazard**
20.	**Object** that *refines* or *is refined by* a **Requirement** is a **Requirement**
21.	**Object** can only be classified as one of **Component**, **Design Definition**, **Hazard**, **Rationale** or **Requirement**
22.	**Object** can only be classified as one of **Assumption**, **Justification** or **Strategy**

3.2 Implementation

This work was conducted using Protégé [37] as a supporting tool for the iterative cycles of specification, implementation and evaluation. The dialect used to specify our ontology is the OWL 2 EL [23], which is sufficiently expressive to represent project data repositories (as discussed in Section 2.3), and the core reasoning is decidable (i.e., the reasoner will not run forever). Also, the use of OWL 2 EL brings some capabilities for inferring implicit classification based on inward and outward relationships, based on a chain of relationships and based on attributes. Finally, complex rules/constraints can be implemented using Semantic Web Rule Language (SWRL), and ontology data can be queried using SPARQL [50]. This technology stack supports many ways to extend the ontology to specific teams' needs.

The entire OWL specification (TBox) for SAO is presented in file `sao.owl` of the available artefacts. We illustrate how the information model (Figure 4) and the rules (Table 3) are translated as OWL specifications in Listing 1. The specification of the relationship *mitigates* and its associated rules read as follows:

- Relationship *mitigates* is implemented in lines 1 to 21 as an object property.
- Rule 13 is implemented in line 2 as a property domain.
- Rule 14 is implemented in lines 3 to 10 as a property range.
- Rule 15 is implemented as a property chain in lines 11 to 14.
- Rule 16 is implemented as a property chain in lines 15 to 18.

Listing 1: OWL specification for *mitigation* relationship and associated rules

```
1  <owl:ObjectProperty rdf:about="omitted:omitted/sao#mitigates">
2    <rdfs:domain rdf:resource="omitted:omitted/sao#↩
         SafetyRequirement"/>
3    <rdfs:range>
4      <owl:Class>
5        <owl:unionOf rdf:parseType="Collection">
6          <rdf:Description rdf:about="omitted:omitted/sao#Cause"/>
7          <rdf:Description rdf:about="omitted:omitted/sao#Hazard↩
               "/>
8        </owl:unionOf>
9      </owl:Class>
10   </rdfs:range>
11   <owl:propertyChainAxiom rdf:parseType="Collection">
12     <rdf:Description rdf:about="omitted:omitted/sao#mitigates"/>
13     <rdf:Description rdf:about="omitted:omitted/sao#causes"/>
14   </owl:propertyChainAxiom>
15   <owl:propertyChainAxiom rdf:parseType="Collection">
16     <rdf:Description rdf:about="omitted:omitted/sao#refines"/>
17     <rdf:Description rdf:about="omitted:omitted/sao#mitigates"/>
18   </owl:propertyChainAxiom>
19   <rdfs:comment>It links a SafetyRequirement to the Hazards that↩
         it addresses</rdfs:comment>
20   <rdfs:label>mitigates</rdfs:label>
21 </owl:ObjectProperty>
```

The loading of data into an OWL ontology requires the preparation of a set of individuals (ABox). For large dataset or to support automation, we need to use machine-readable format. Thus, the datasets used to evaluate SAO follow the RDF Turtle syntax [7]. Turtle allows the specification of a data graph in a compact and natural text form. In this way, Turtle syntax is both machine-readable and human-readable and can be handled by many third-party tools and libraries. Regarding the semantic schema, the format uses the SAO ontology and the RDF Schema [8]. Listing 2 shows an example of project data represented in the interchange format.

Listing 2: Example of data represented using SAO

```
1 :R1 rdf:type sao:Requirement .
2 :R1 rdfs:label "Requirement␣1" .
3 :R1 rdfs:comment "This␣is␣a␣requirement" .
4 :H2 rdf:type sao:Hazard .
5 :H2 rdfs:label "Hazard␣2" .
6 :R1 sao:mitigates :H2 .
```

This data format serves as an integration point between SAO and any project repository. Instead of extending the core implementation of the framework to the specifics of each management tool, the responsibility of transforming the project data into the interchange format is delegated.

We expect variability in how teams and companies represent and manage their development data inside project management tools [3]. Imposing a rigid information model may hinder the adoption of any proposed solution.

Thus, teams may use different tools such as Issue Tracking System (ITS) or Application Lifecycle Management (ALM), but the concepts and relationships underlying their project data have to comply with regulations. Different information retrievers will need to be implemented based on the exporting interfaces of each tool. In another scenario, teams may use the same tool in different ways. For example, let two teams use an ITS, Team A chooses to represent

hazards as a customised issue type, while Team B chooses to represent them as an issue which contains a label "hazard". We need two different information retrievers to extract this type of data from these two settings.

The evaluation of an ontology on the implementation level seeks to verify if the specification is consistent [6]. There are three steps to verify an ontology implementation:

(1) If the set of axioms (TBox) is consistent;
(2) If the set of axioms combined with valid individuals (ABox) is consistent;
(3) If the set of axioms combined with invalid individuals is inconsistent.

The Protégé tool provides support for all these types of verification. Ontology engineers can specify axioms and assertions (known as individuals) inside the editor and also trigger the execution of OWL reasoners. In steps 1 and 2, the expected result is that no inconsistency is found. In step 3, the expected result is that the reasoner returns an inconsistency. Step 3 is intricate as the engineer needs to elaborate assertions for each aspect of the axioms that will be tested [43].

For example, to verify the proper implementation of Rule 13 (Table 3), we need to feed our ontology with some data items and check if the items are correctly classified. The following project data can verify the rule:

- X is a project item of any type (except "SafetyRequirement");
- Y is another project item of any type;
- Y mitigates X.
- Expected result after reasoning: X is classified as Safety Requirement.

SAO is currently incorporated in an assurance framework that the authors designed and developed [4]. The reference implementation of the framework is implemented as a web system that provides a user interface to demonstrate its features. It can retrieve data automatically from an issue tracker or load data from files using the data format based in Turtle RDF. It automates data retrieval, loading, reasoning, querying of competency questions, presentation of the project data, and generation of assurance cases fragments in multiple argumentation patterns. The framework aims to support teams in continuously evaluating the project information quality and traceability. In this context, the ontology is vital to allow teams to work together in the same tools, rather than in separate, specialised tools (e.g., requirements, hazard analysis, etc.).

No false positives nor false negatives from the reasoner were noticed. Empty results were quickly checked and attributed to missing individuals or relationships in the dataset, not inference faults. As a quick check, we tweaked one of the datasets with invalid statements to assert that incoherent items would trigger an inconsistency, not a silent misclassification. For example, adding a *mitigates* link between two Hazards would infer that a Hazard is a SafetyRequirement, which conflicts with it already being a Hazard (those classes are disjoint). Such inconsistency was promptly identified by the reasoner.

We adapted two datasets for use during the ontology implementation and evaluation with experts (see Section *Artefacts Availability*). Original datasets comprise requirements, hazards, and traceability information available in different formats (e.g., structured reports, GSN, JSON, Spreadsheets). To feed such datasets into SAO, data was manually restructured into RDF triples. (see Q1.1 results).

4 Evaluation

We evaluated the SAO through illustrative scenario and an expert survey with practitioners.

4.1 Illustrative Scenario with Public Datasets

Although there are publicly available requirements datasets and examples of ACs, there are few datasets comprising safety requirements and the associated ACs. We discuss two public datasets below that were used to evaluate our framework.

Open Patient-Controlled Analgesia Pump Project. The Open PCA Pump Project (OpenPCA) is an open-source, safety-critical medical device project developed to provide a public resource for researchers and practitioners [26]. The project illustrates a suite of realistic development artefacts comprising requirements, architecture, source code, tests, risk management, and ACs.

The dataset comprised the following provided and inferred data metrics:

- Items provided: 94
- Item links provided: 203
- Item attributes provided: 266
- New classifications for items inferred: 49
- New item links inferred: 105

For all Basic competency questions, SAO found at least one item as the answer. For the Traceability Gap Questions, SAO found items with possible inconsistencies in questions CQ14, CQ15, CQ16, CQ17, CQ20, CQ21, and CQ24.

Dronology Project. This project provides an open-source framework for controlling and coordinating the flight of individual Unmanned Aircraft Systems, formations, and swarms to support applications such as search-and-rescue, surveillance, and scientific data collection. The project establishes a research environment for studying various software and systems engineering aspects for cyber-physical systems -– for example, runtime monitoring, safety analysis, and product line development [13].

The dataset comprised the following provided and inferred data metrics:

- Items provided: 155
- Item links provided: 334
- Item attributes provided: 552
- New classifications for items inferred: 64
- New item links inferred: 22

For the following Basic Questions, SAO did not find any items (empty answer): CQ07, CQ11, CQ12 and CQ13. For the following Traceability Gap Questions, SAO found items with possible inconsistencies: CQ14, CQ16, CQ17, CQ20, CQ21, and CQ22. Empty answers reflect missing individuals/links in the dataset, not added or altered semantics. As no items of type Causes, Strategy, Justification nor Sources are available in this project, SAO promptly detected inconsistencies.

SAO's metamodel encompasses all the concepts and relationships available in these two public datasets. Once the project information

is modelled into SAO, users can infer implicit information and highlight traceability gaps in these datasets.

4.2 Expert Survey

We also conducted a study to evaluate the ontology from the point of view of industrial practitioners and researchers. We ran a survey research [18] based on a self-administered questionnaire [32]. The objective was to gather the perception of experts about the utility of the ontology.

4.2.1 Survey design and sampling. We defined the following research questions to guide the evaluation:

- **Q1**. How useful is the SAO information model to represent core concepts/relationships related to assurance and project information?
- **Q2**. How useful are the SAO's competency questions?
- **Q3**. Could the use of SAO reduce the effort to retrieve and assess safety assurance information?

This expert survey's target population was researchers and professionals involved in developing critical systems and with any experience/knowledge about SCS or ACs. The unit of analysis for the survey sample is the practitioner who works on SCS projects. The survey participation is individual and anonymous. We used convenience sampling as a strategy with our collaboration network. We sent an invitation by email to our contacts explaining the evaluation objectives and how to perform it.

Seven professionals participated in the evaluation, and their profiles are depicted in Table 4. Their roles are: P1 is a researcher, P2 is a senior software specialist, P3 is an engineer, P4 is a senior design assurance and QA engineer, P5 is an embedded software developer, P6 is a functional safety QA tester, and P7 is a researcher. Besides the short sample, they are very representative of the target population. Five participants have more than 10 years of experience in SCS projects. They represent the target public, which are people involved in SCS development and acquainted with project management tools, but not necessarily knowledgeable of AC development.

Survey questions were derived from SAO motivations and expected benefits, such as traceability, consistency analysis, and knowledge discovery. A pilot application was conducted with two participants. One has large development industry experience in many fields, and recently in the SCS domain, while the other has solid experience in RE research. Feedback from this preliminary validation was incorporated into the final survey design (e.g., refine clarity/timing).

Table 4: Participants' profile

Question	P1	P2	P3	P4	P5	P6	P7
Experience in SCS (years)	+10	+10	6–10	+10	3–5	+10	+10
No. of projects using ACs	+10	None	1–4	None	None	+10	+10
Familiarity with technologies/tools							
Assurance Case	2	0	1	0	-2	0	2
GSN notation	2	-2	-2	2	-2	0	2
Ontology specification	2	1	-1	-2	-2	-1	1
Description logic	0	1	-2	-2	-2	1	1

Extremely familiar=2, Moderately familiar=1, Somewhat familiar=0, Slightly familiar=-1, Not at all familiar=-2.

For the final survey design, participants were trained through short explanatory videos and written instructions that introduced SAO metamodel, its main classes and the competency questions. We also asked them to navigate through the competency questions answers generated by SAO for each of the datasets. After, they answered opinion questions to collect their judgement on the ontology's features. There were no open questions for reviewing the ontology; instead, questions were linked to participants' professional experience (e.g., Q1.1 and Q1.2).

4.2.2 Q1 – How useful is the SAO information model to represent core concepts/relationships related to assurance and project information? The sub-questions for Q1 are listed in Table 5. These questions aim to gather evidence on the utility of the SAO in terms of terminology coverage. Table 6 shows the answers for opinion question Q1.1.

Table 5: Sub-questions for Q1

Description	Answer type
Q1.1: For each concept of the Safety Assurance Information Model, please indicate if it applies to the projects you participated in the last 5 years (for each concept, a frequency is asked)	Frequency
Q1.2: From your experience, please indicate any important concept which is not covered in the Safety Assurance Information Model	Open-ended

Table 6: Frequency of concept usage (Q1.1)

Concept	P1	P2	P3	P4	P5	P6	P7	Average
Requirement	2	2	0	2	2	2	2	1.7
Safety Requirement	2	2	-2	2	-1	2	1	.85
Hazard	2	2	-2	2	-2	2	1	.7
Cause (of Hazard)	2	2	-2	2	-2	2	1	.7
Design Definition	2	2	1	2	0	2	2	1.6
Component	2	2	?	2	-2	2	2	1.3
Context	2	1	?	2	2	2	1	1.7
Assumption	1	2	1	2	-2	2	2	1.1
Justification	1	2	1	2	0	2	2	1.4
Strategy	0	1	1	?	-2	1	0	.2
Object	2	1	?	2	-2	1	2	1
Source	2	1	?	2	-2	1	2	1

Never used=-2, Few projects=-1, Some projects=0, Most of the projects=1, Every project=2, Don't know=?.

From the answers in Table 6, we observed that P3 and P5's answers deviate from the others. They either work in very specific niches in the development process or are familiar with another terminology. Even after going through the short lectures and the tasks, P3 and P5 could not establish a mapping among SAO concepts and those of their work experience. Their roles are engineer and developer, and the gap between safety engineering and software engineering may explain the mismatch [27].

From P1, P2, P4, P6 and P7 answers, we observed some high frequency of use of fundamental concepts such as Requirement, Safety Requirement, Hazard, Cause, Design Definition and Component. This is expected, as these concepts appear in various regulations [48]. The terms we brought from AC domain (Context, Assumption, Justification, Strategy) have mixed frequencies reported, which may relate to the fact that ACs are not demanded by all regulations.

In question Q1.2, we asked for concepts not covered in the ontology. One participant pointed out that SAO's focus is on the early requirements and development process, but as safety and assurance engineering pervades the entire product lifecycle, the model should cover more concepts related to evidence types and other kind of artefacts such as "*lifecycle plan*". In fact, the SAO's information model currently handles any kind of evidence or external artefacts just as *Source*. Another participant indicated the concept of "*Operation*" as important to make the SAO's information model more complete. This information can be managed as a special kind of "Context", but this was not clear to the participant. Two participants did not indicate any missing concept.

Discussion for Q1. Our information model was designed to be lean and be extended for application in real-world settings. We observed that the concepts in the current version of SAO are useful.

Regarding important concepts not covered, we expected to receive more specific indications of missing concepts. One of the participants indicated a category of concept (evidence types), for which in a recent literature survey more than 40 different types of evidence were found [38]. Another participant indicated the concept of operation, for which the Safe-RE metamodel [48] provides the enumeration "Context Type" (with values *Operational* and *Environmental*) to be used as an attribute in "Context" instances.

Bringing all these types and enumerations suggested by the participant may inflate the model. However, the inclusion of some evidence types (the most common) serves to show how the SAO can handle specific needs in a real-world setting. It is worth noting that all missing concepts reported by experts are specializations of core concepts of the ontology and do not compromise SAO core coverage as the ontology already models essential classes and relations needed to support the competency questions. Practitioners pointed to these concepts as an opportunity for extensions and further evolution.

The deviation in the frequency of use of the concepts observed among one participant compared to the others is a good example of the challenge of bridging different development areas towards a single repository. In spite of years of experience in safety-critical systems (SCS), practitioners may work in specific roles or parts of a project that may either not manage some kind of artefacts or use different terminology. The adoption of ontology technology can handle this challenge, as we can extend the ontology specification to include different terminologies for the same concept/relationships.

4.2.3 Q2 – How useful are the SAO's competency questions? The sub-questions for Q2 are listed in Table 7. These questions aim to gather evidence on the utility of the SAO's competency questions and the consistency analysis that is inherent to ontologies designed using description logics. For each questions, we asked for additional comments.

For Q2.1, there were four "To a great extent", two "Somewhat" and one "Very little" answers for the relevance of the basic competency questions. For Q2.2, there were five "To a great extent", and two "Somewhat" answers regarding the relevance of the traceability gaps competency questions. Finally, for Q2.3, there were four "To a great extent", and three "Somewhat" answers related to the relevance of the SAO capability to check inconsistencies.

Table 7: Sub-questions for Q2

Description	Answer type
Q2.1: To what extent do you find relevant the "Basic" competency questions that organises all project items by their types/classifications?	Relevance
Q2.2: To what extent do you find relevant the "Traceability Gaps" competency questions that reveals gaps in traceability?	Relevance
Q2.3: To what extent do you find relevant the SAO feature that checks consistency and generates explanations in case of inconsistent project data?	Relevance

Relevance scale: Not at all, Very Little, Somewhat, To a Great Extent.

Discussion for Q2. We observed that SAO ontology was perceived as useful by the participants. It is worth noting that participants do not have much familiarity with description logic (see Table 4), so we can consider their opinion as a final user perspective.

For question Q2.1, we received the comment: "*It helps to reunite all the necessary information necessary to perform the analysis and identify the gaps that must be covered, in a very easy and logical way.*" For question 2.2, we received the comment: "*This helps ensure that no gaps will be missed by the engineer, it is really a supporting tool and not something to automate the analysis.*" Both comments are aligned with the objectives of SAO.

For question Q2.3, we received the comment: "*Most of/all the checks would not be needed if model instances were created with a tool that ensures correctness-by-construction.*" Truly, a model-based approach for assurance information management could ensure correctness during the construction of model instances [15]. However, a motivation of our work is to investigate how we could bring the support of consistency analysis to project data repositories in which we cannot (or do not want to) enforce constraints in the create/update actions.

4.2.4 Q3 – Does the use of SAO could reduce the effort to retrieve and assess safety assurance information? The opinion questions for this ES-RQ are listed in Table 8. These questions aim to gather evidence on the perceived likelihood that the adoption of SAO could offer the expected benefits. Table 9 shows the answers to these opinion questions.

Table 8: Question for Q3

Description	Answer type
To what extent do you believe the adoption of the SAO could reduce the effort to review safety traceability information?	Likelihood

Table 9: Participants' answers for Q3

Question	P1	P2	P3	P4	P5	P6	P7
Likelihood to reduce traceability review effort	3	3	3	4	3	3	1

Likelihood scale: Not at all=1, Very Little=2, Somewhat=3, To a Great Extent=4.

Discussion for Q3. From the answers in Table 9, we observed that SAO was perceived as beneficial by the participants, but not to a great extent. We received the comment "*It can be used to support a requirements review but, unless it is qualified, the reviewer cannot rely on the information provided.*" Another comment was

"*The navigation is not very intuitive and there is a lot of information visible on the main page.*", which is related to the extend output of the competency questions. As to address those issues, the process of extending SAO for utilization in a real-world setting needs to fine-tune the information model and rules, validate the semantics of the traceability with project stakeholders, and also seek ways to better present traceability information for end users.

4.3 Threats to Validity

This section discusses the threats to validity concerning our expert survey's planning, design, and execution. We adopted the approach presented by Wohlin et al. [51].

Construct validity. In a survey, the quality of the questions ensures that the instrument measures what it intends to measure. As we adopted TAM, our main questions aimed to collect opinions. For each single feature of the ontology, we co-designed tasks and opinion questions. We interleaved tasks and questions throughout the procedure so that participants could clearly understand for what they were providing opinions.

One limitation is that all tasks correspond to information retrieval activity within the context of a development project. We consider that tasks for creating or updating information could be lengthy or require extended training. A case study would be a more appropriate kind of study to run a comprehensive evaluation of the SAO ontology.

External validity. This survey was designed to evaluate our ontology, and it is not meant to generalise claims over the target population. However, it serves as a initial evaluation of the solution in the eyes of a highly skilled and experienced group of practitioners. As the instrument was designed to be self-administered, its replication is feasible.

Internal validity. The selection of participants and how to treat their answers may affect the internal validity of a survey. To avoid responses from people outside the expected profile, we clearly explained the desired profile in the invitation. We also included personal profile questions towards the experience with the many aspects involved in our solution: SCS development, AC development, ontology specification, description logics and issue trackers. The participants' general profile comprises professionals with considerable expertise in SCS development and AC development.

5 Conclusions and Future Work

The main contribution of this work is the development of an ontology to support the representation, assessment and classification of assurance and regular project information. SAO comprises an information model, constraints, rules of formations, and competency questions. The main purpose is to support teams in continuously evaluating project information quality and traceability. Our ontology provides a lean information model and set of rules that can be applied to early stages of safety requirements development.

The ontology was evaluated through an illustrative scenario with public datasets, and an expert survey with seven practitioners. We observed that SAO can support early and continuous detection of gaps, automatic classification of items, and AC generation. During the practitioner's evaluation, we gathered evidence that i) competency questions on traceability gaps are useful in revealing direct and indirect relationships among project items, ii) basic competency questions facilitate the contextualisation of items traced in the project repositories, iii) the inclusion of specialisations for some concepts may enhance practitioners' grasp of the metamodel scope.

SAO's functionalities can be used by practitioners to reduce errors, cost, and rework during certification audits, where corrections are much more costly. These arguments support SAO's potential to positively impact software development processes, although empirical evidence from real projects is still required to confirm such outcomes. As SAO is designed to be lean, we could only evaluate the extensibility of its information model, specification, and competency questions when used in a real-world setting.

In the wake of the next generation of AI-based approaches relying on machine learning and large language models, we see the lack of public datasets and the limited size of the available ones as challenging constraints to apply generative AI approaches to assess safety assurance information from a dataset comprising safety requirements, hazard analysis, and system design/implementation.

Practitioners developing safety-critical systems and producing ACs lack proper tools to manage and review assurance information and to map them to AC arguments. As future work, we aim to implement a framework to support teams with a collaborative place where the interrelations between assurance and regular project information can be explicitly managed together early and continuously throughout the system development lifecycle. SAO is a central piece of this framework, which leverages early and continuous quality assessment of assurance information recorded in project repositories, as well as the automated generation of ACs. SAO can be used with or without the associated framework.

Future research involves assessing performance and scalability, as SAO underlying reasoning process (OWL) can be a bottleneck on large-scale graphs. The two public datasets can be processed in less than one minute each. Therefore, only just-in-time response scenarios are constrained in SAO.

We also consider it important to extend the ontology to encompass more sub-types for classes such as Solution (Evidence), Context, Strategy and Requirements. Such extension brings more readiness to adopt the ontology and favours the understanding of the information model concepts and relationships.

ARTIFACT AVAILABILITY

Artifacts related to this work are available at https://github.com/arcade-framework/datasets-queries. The artifacts include the complete SAO specification, two datasets adapted from open projects discussed in Section 4, a dataset with some invalid statements, and the implementation using SPARQL of the queries that extract answers for the competency questions.

ACKNOWLEDGMENTS

This work was partially supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

REFERENCES

[1] Ankit Agrawal, Seyedehzahra Khoshmanesh, Michael Vierhauser, Mona Rahimi, Jane Cleland-Huang, and Robyn Lutz. 2019. Leveraging artifact trees to evolve

and reuse safety cases. In *Proc. of the 41st Intl. Conf. on Software Eng*. IEEE Press, 1222–1233.

[2] Camilo Almendra, Flavia Barros, and Carla Silva. 2019. Using Assurance Cases in Requirements Engineering for Safety-Critical Systems. In *Anais Estendidos da X Conferência Brasileira de Software: Teoria e Prática* (Salvador). SBC, Porto Alegre, RS, Brasil, 47–55.

[3] Camilo Almendra and Carla Silva. 2020. Managing Assurance Information: A Solution Based on Issue Tracking Systems. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (Natal, Brazil) *(SBES '20)*. Association for Computing Machinery, New York, NY, USA, 580–585. doi:10.1145/3422392.3422454

[4] Camilo Almendra and Carla Silva. 2023. ARCADE: A Framework for Integrated Management of Safety Assurance Information. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*.

[5] Eric Armengaud. 2014. Automated safety case compilation for product-based argumentation. In *Embedded Real Time Software and Systems (ERTS2014)*.

[6] Samantha Bail. 2013. Common reasons for ontology inconsistency. https://ontogenesis.knowledgeblog.org/1343/

[7] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. 2014. *RDF 1.1 Turtle - Terse RDF Triple Language*. Technical Report. World Wide Web Consortium (W3C). https://www.w3.org/TR/2014/REC-turtle-20140225/

[8] Dan Brickley and R.V. Guha. 2014. *RDF Schema 1.1*. Technical Report. World Wide Web Consortium. https://www.w3.org/TR/rdf11-schema/

[9] Janet E. Burge, John M. Carroll, Raymond McCall, and Ivan Mistrik. 2008. *Rationale and Requirements Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 139–153.

[10] CENELEC EN 50129:2018 2018. *CENELEC EN 50129 – Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling*. Standard. European Committee for Electrotechnical Standardization (CENELEC), Belgium.

[11] Jane Cleland-Huang, Orlena C. Z. Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. 2014. Software Traceability: Trends and Future Directions. In *Proceedings of the on Future of Software Engineering* (Hyderabad, India) *(FOSE 2014)*. ACM, New York, NY, USA, 55–69.

[12] J. Cleland-Huang and M. Vierhauser. 2018. Discovering, Analyzing, and Managing Safety Stories in Agile Projects. *IEEE 26th Intl. Requirements Engineering Conf.* (2018), 262–273.

[13] Jane Cleland-Huang, Michael Vierhauser, and Sean Bayley. 2018. Dronology: An Incubator for Cyber-Physical Systems Research. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results* (Gothenburg, Sweden) *(ICSE-NIER '18)*. Association for Computing Machinery, New York, NY, USA, 109–112. doi:10.1145/3183399.3183408

[14] J.L. De La Vara, A. Ruiz, and H. Espinoza. 2018. Recent Advances towards the Industrial Application of Model-Driven Engineering for Assurance of Safety-Critical Systems. In *Proc. of the 6th Intl. Conf. on Model-Driven Engineering and Software Development (MODELSWARD)*. 632–641.

[15] Jose Luis De La Vara, Arturo S García, Jorge Valero, and Clara Ayora. 2022. Model-based assurance evidence management for safety–critical systems. *Software and Systems Modeling* 21, 6 (2022), 2329–2365.

[16] Ewen Denney and Ganesh Pai. 2012. A lightweight methodology for safety case assembly. In *Computer Safety, Reliability, and Security: 31st International Conference, SAFECOMP 2012, Magdeburg, Germany, September 25-28, 2012. Proceedings 31*. Springer, 1–12.

[17] D. Dermeval, J. Vilela, I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva. 2016. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering* 21, 4 (01 Nov 2016), 405–437.

[18] S. Easterbrook, J. Singer, M. Storey, and D. Damian. 2008. *Selecting Empirical Methods for Software Engineering Research*. Springer, London, 285–311.

[19] Eclipse Foundation. 2023. Eclipse OpenCert. https://www.polarsys.org/opencert/

[20] FDA 2018 2014. *FDA — Infusion Pumps Total Product Life Cycle – Guidance for Industry and FDA Staff*. Guidance. Food and Drug Administration, USA.

[21] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. 1997. METHONTOLOGY: from ontological art towards ontological engineering. In *Proc. AAAI Spring Symposium*. American Association for Artificial Intelligence, 33–40.

[22] Generic Infusion Pump Research Project. 2023. The Generic Infusion Pump (GIP) - A workbench for improving safety, security and usability of medical systems. https://rtg.cis.upenn.edu/gip/

[23] Christine Golbreich and Evan K. Wallace. 2012. *OWL 2 Web Ontology Language New Features and Rationale*. Technical Report. World Wide Web Consortium. https://www.w3.org/TR/owl2-new-features/#OWL_2_EL

[24] Asunción Gómez-Pérez and Richard Benjamins. 1999. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings.

[25] GSN v3 2021. *Goal Structuring Notation Community Standard (Version 3)*. Standard. The Assurance Case Working Group. https://scsc.uk/r141C

[26] John Hatcliff, Brian Larson, Todd Carpenter, Paul Jones, Yi Zhang, and Joseph Jorgens. 2018. The Open PCA pump project: an exemplar open source medical device as a community resource. In *Proceedings of the 2018 Medical Cyber-Physical Systems (MedCPS) Workshop*.

[27] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones. 2014. Certifiably Safe Software-dependent Systems: Challenges and Directions. In *Proc. of the on Future of Software Engineering (FOSE 2014)*. ACM, India, 182–200.

[28] ISO 26262:2018 2018. *ISO 26262 – Road vehicles – Functional safety – Part 10*. Standard. ISO, Switzerland.

[29] ISO/AWI TS 81001-2-1 2021. *ISO/AWI TS 81001-2-1 – Health software and health IT systems safety, effectiveness and security – Part 2-1: Coordination – Guidance for the use of assurance cases for safety and security*. Technical Specification.

[30] ISO/IEC/IEEE 15026-2 2022. *ISO/IEC/IEEE 15026-2 – Systems and software engineering – Systems and software assurance – Part 2: Assurance case*. Standard.

[31] Tim Kelly. 2018. *Safety Cases*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 361–385. doi:10.1002/9781119443070.ch16

[32] Barbara A. Kitchenham and Shari L. Pfleeger. 2008. Personal Opinion Surveys. In *Guide to Advanced Empirical Software Engineering*. Springer London, London, 63–92. doi:10.1007/978-1-84800-044-5_3

[33] Brian Larson. 2014. *Open PCA Pump Assurance Case*. Technical Report. The University of York, York. http://santoslab.org/pub/open-pca-pump/artifacts/Open-PCA-Pump-Safety-Case.pdf Not available. Last accessed: 2019-08-07.

[34] Brian R. Larson and John Hatlicff. 2018. *Open Patient-Controlled Analgesia Infusion Pump System Requirements 1.0.0*. Technical Report. Kansas State University, Kansas State University. https://web.archive.org/web/20160312145247/http://santoslab.org/pub/open-pca-pump/artifacts/Open-PCA-Pump-Requirements.pdf Accessed: 2019-08-07.

[35] P. Mäder, P. Jones, Y. Zhang, and J. Cleland-Huang. 2013. Strategic Traceability for Safety-Critical Projects. *IEEE Software* 30, 3 (May 2013), 58–66.

[36] MoD DefStan 00-056:2017 2017. *Defence Standard 00-056 – Safety Management Requirements for Defence Systems – Part 2 – Guidance on establishing a means of complying with part 1*. Standard. Ministry of Defence, UK.

[37] Mark A. Musen. 2015. The protégé project: a look back and a look forward. *AI Matters* 1, 4 (2015), 4–12. doi:10.1145/2757001.2757003

[38] Sunil Nair, Jose Luis de la Vara, Mehrdad Sabetzadeh, and Lionel Briand. 2014. An extended systematic literature review on provision of evidence for safety certification. *Information and Software Technology* 56, 7 (jul 2014), 689–717. doi:10.1016/j.infsof.2014.03.001

[39] Ignazio Palmisano. 2023. OWL API main repository. https://github.com/owlcs/owlapi

[40] B. Ramesh and M. Jarke. 2001. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering* 27, 1 (Jan 2001), 58–93.

[41] Patricia Rodríguez-dapena. 1999. Software safety certification: a multidomain problem. *IEEE Software* 16, 4 (1999), 31–38.

[42] J. Rushby. 2015. The interpretation and evaluation of assurance cases. *Comp. Science Laboratory, SRI Intl., Tech. Rep. SRI-CSL-15-01* (2015).

[43] Konstantin Schekotihin, Patrick Rodler, Wolfgang Schmid, Matthew Horridge, and Tania Tudorache. 2018. Test-Driven Ontology Development in Protégé.. In *ICBO*.

[44] Guus Schreiber. 2008. Knowledge engineering. In *Handbook of Knowledge Representation*. Elsevier, Chapter 25, 929–946.

[45] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Boris Villazón-Terrazas. 2009. *How to Write and Use the Ontology Requirements Specification Document*. Springer Berlin Heidelberg, Berlin, Heidelberg, 966–982.

[46] J. Valaski, S. Reinehr, and A. Malucelli. 2016. Which Roles Ontologies play on Software Requirements Engineering? A Systematic Review. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens, 24–30.

[47] Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, and Tony Gorschek. 2017. Integration between requirements engineering and safety analysis: A systematic literature review. *Journal of Systems and Software* 125 (2017), 68 – 92.

[48] Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G Martins, and Tony Gorschek. 2018. Safe-RE: a safety requirements metamodel based on industry safety standards. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering*. ACM, 196–201.

[49] W3C. 2023. Resource Description Framework (RDF). https://www.w3.org/RDF/

[50] W3C. 2023. Semantic Web Standards. https://www.w3.org/2001/sw/wiki/Main_Page

[51] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. doi:10.1007/978-3-642-29044-2