ARCADE: a Framework for Integrated Management of Safety Assurance Information

Camilo Almendra
Campus de Quixadá
Universidade Federal do Ceará
Quixadá, Brazil
camilo.almendra@ufc.br

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

***Abstract*—Assurance Case Development is an emerging approach for demonstrating that a system is safe. An assurance case includes top-level claims for a property of a system (e.g., safety) and a structured argumentation that breaks down claims into supporting evidence. Its production is an effortful activity that relies on project information and its associated traceability. It is challenging for developers to build assurance cases as they have to identify and gather the project information relevant to the argumentation and assess its consistency and completeness. However, regular project information (e.g. requirements, design, rationale) and assurance-related information (e.g. hazard, causes, mitigation strategies) are likely managed separately. In this context, we designed the ARCADE framework to support the integration of assurance information together with regular information inside project management tools, and to perform automated consistency/completeness analysis and generation of assurance cases. Thus, our approach promotes early and continuous review of traceability information needed to produce assurance cases, while automating their construction. In this work, we present the design of the framework and its current implementation.**

I. Introduction

A certification procedure is the last stage in the development lifecycle in regulated domains before releasing a safety-critical system for operation [1]. Regulations are built and evolved toward a body of knowledge gathered from domain experts and the systems' field operation. Traditionally, regulations follow only prescriptive approaches. They comprise obligatory development practices and explicitly require the submission of documentation that communicates how process and product evidence demonstrate the safety of the software functions [1]. On the other hand, goal-based approaches, also called evidence-based certification, rely on the statement of high-level objectives and require vendors to submit an argument that communicates how product evidence supports the objectives concerning the safety of the software [1].

Assurance case development is a trending goal-based approach to communicate and argue over the safety of a system, offering more flexibility to justify requirements and design choices aimed at mitigating risks and hazards [2]. The purpose of an assurance case is to present a case in favor of a system. ISO/IEC 15026–2 states that *an assurance case includes a top-level claim for a property of a system or product (or set of claims), systematic argumentation regarding this claim, and the evidence and explicit assumptions that underlie this argumentation.* [3]. Assurance case arguments are hierarchical, beginning with a top claim that is decomposed into sub-claims or satisfied by evidence. Providing evidence to back up claims is the fundamental objective of assurance argument development.

The building blocks of assurance arguments are the requirements, design decisions, hazards, safety requirements, mitigation strategies and all other project information relevant to substantiated that the system will perform as intended and without unacceptable risks. In a safety-critical project, lack of integration between safety and software development teams impacts the quality of assurance cases [4]. Practitioners report that the lack of integration between assurance case development and system development lifecycles makes it harder to handle changes, discover and organize information that should be present in the assurance cases [5].

Cross-collaboration between the safety assurance team and the requirements engineering team is key. Collaboration towards safety and requirements activities is reported in industry practice [6]. Combination of assurance case development and requirements activities is also reported in practice, yet it is not widespread and lacks proper tool support [7].

The development of assurance cases along with requirements engineering could foster early discussion of safety concerns, enable the system design to be reviewed from a safety assurance view, and bridge the expertise of various stakeholders involved in safety-critical systems development. We envision assurance and software teams working together in an integrated repository where each team can manage the project artifacts under their responsibility. To

achieve this, we proposed the ARCADE (AssuRance CAse DEvelopment) framework and discussed its design in this work.

The contributions of this work are:

- the Safety Assurance Information Model that integrates safety assurance and regular project information and can be adopted as underlying information model for customization of project management tools;
- the Safety Assurance Ontology that models the informational needs of teams which develop safety-critical systems and assurance cases;
- and the ARCADE framework implementation that operationalizes the information model and the ontology to provide automated consistency/completeness analysis and generation of assurance cases.

This paper is structured as follows. Section II discusses the background and related works. Section III presents an overview of the framework. In Section IV, the components and features of the framework are presented. Finally, we discuss conclusions and future work in Section V.

II. Background

Assurance Case Development. Nowadays, some regulations mandate or recommend elaborating an assurance case as part of the certification documentation. Examples of regulations are the standards ISO 26262 for road vehicles, EN 50129 for railway applications, the Def Stan 00-56 for defense systems, and the FDA guidance for infusion pumps. The standard ISO/IEC 15026 defines common terminology and specifies the general structure, contents, and life cycle for assurance cases, while there are no restrictions upon notation and tools [3]. Moreover, there are initiatives developing new regulations towards assurance cases, such as the ISO/AWI TS 81001-2-1 on the health software and IT domain [8].

They comprise an amalgamation of information from standard and product requirements, risk and hazard analysis results, design decisions and rationale, validation and verification results, and process management records [2]. Such information is organized in *argument structures*, comprising high-level requirements and goals of the system, evidence items that support the satisfaction of requirements and goals, and arguments that explain and justify this support from the evidence items [2].

A recurring problem is the difficulty for teams to build complete, correct, and compelling arguments [1]. When developing assurance arguments, practitioners face a lack of expertise and guidance in identifying relevant information and constructing the arguments [5], [9].

Although assurance cases require much information that will only be completely available at the end of the development lifecycle, the practice recommends elaborating on them in the early phases of development [2], [9]. Postponing the creation of assurance cases can lead to late rework due to the impossibility of building a compelling assurance case when the collected evidence and arguments are not sufficient to hold the top-level claims [2]. Furthermore, teams may also experience the late discovery of safety requirements or constraints during assurance case development activities [5].

Assurance Information Management. The central spine of safety arguments are requirements, hazards, design, and the rationale to justify the decisions. Thus, creating assurance cases demands an explicit statement of assumptions, justifications, and argumentation strategies [10]. Rationale represents one of the information objects traced during development, alongside requirements, assumptions, designs, components, decisions, and others [11]. When developing safety-critical system, rationale provides the confidence to assure that requirements decomposition and design decisions reasonably address all identified risks and hazards. Along with specifications, rationale must be propagated throughout the development process [10]. Traceability management is strategic to assure that software is safe for use. Many regulators of various industry domains have incorporated it into standards and guidelines [12].

Integrated project information management. Current project management tools provide features to query and extract project data. In particular, issue tracking systems are a widely used project management tool that records much knowledge about the system under development. Each tracking system has its schema for representing project assets, and many allow users to create customized concepts and relationships among them. So, any approach that aims to be practical for industry application has to support different ways to organize project data that we could encounter in real project settings.

Our target scenario is any team/company that already manages regular project information inside a tracking system or similar tool, thus requirements, design definitions, tests, and other project items are somehow managed inside a tool. Making teams manage safety assurance information using their well-established, daily-used project management tool may leverage a system's safety analysis and the development of assurance cases [10].

III. ARCADE Framework

Our vision is to support the integrated management of assurance information together with regular information directly into project management tools.

Figure 1 shows how ARCADE leverages the assurance case development in the target scenario. A key component is the Safety Assurance Information Model that combines fundamental project items (e.g., requirements, design, tests) with safety-related project items (e.g., hazards/risk analysis, safety requirements, justifications, assumptions). This information model serves as an underlying traceability schema and guides the customization of management tools for a specific project or company. Adopting the model alone enables a unified place where software and safety teams can work and maintain traceability together.

The framework provides a base implementation of a project information retriever. It implements the extraction of project items from a popular issue tracking system [1] and transforms

[1]Atlassian Jira Software - https://www.atlassian.com/software/jira

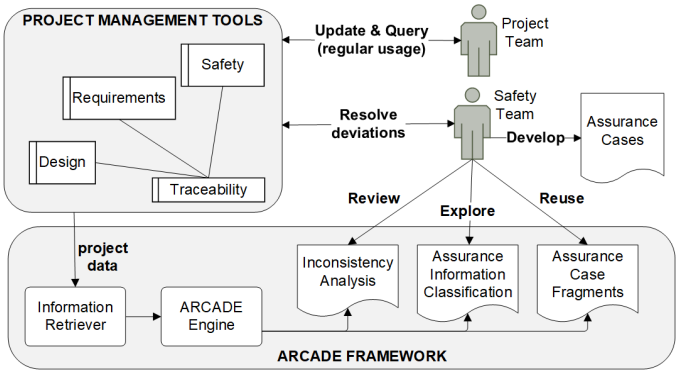

Fig. 1. Overview of assurance case development supported by ARCADE.

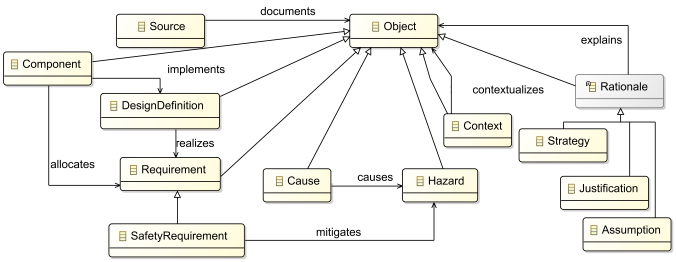

Fig. 2. Safety assurance information model

them to the interchange format of ARCADE (example of the format available in the supplementary material [13]). Users can extend the tool to support extracting data from other project management tools.

The main component of the framework is the engine which embeds an operational ontology, an assurance case generator, and a web user interface. The engine receives the data specified in the interchange format, performs consistency and completeness analysis, and generates assurance case fragments. Users can extend the engine to enforce particular constraints/rules for specific projects. The design of the framework also supports extending the assurance case fragments generator. Thus, the framework's design provides a reference implementation that could be readily adapted for particular projects' settings, yet providing a helpful base implementation.

The goals of ARCADE framework are: **(G1)** Support the integrated management of assurance information together with regular; **(G2)** Support the automated assessment of assurance-related and regular information; **(G3)** Support the automated generation of assurance case fragments; **(G4)** Support the customization of the framework for its adoption in different safety-critical system project settings.

IV. Framework Design

ARCADE comprises four main components:

- an information model used as the schema for integration of safety assurance information into the management tools;
- an ontology built on top of the information model that embodies the knowledge managed by the framework;
- a project information retriever that extracts data from management tools; and
- the main engine of the framework that runs the data analysis, provides data visualization and generates assurance case fragments.

A. Safety Assurance Information Model

The information model serves as the basis for the management of assurance rationale information inside a tracking system, thus leveraging the development of assurance cases. Adopting an underlying information model is fundamental to supporting certification procedures [12]. The information model supports modeling requirements, hazards, design components, assurance rationale, and evidence items (Fig. 2). We based our model on three reference metamodels: Ramesh and Jarke's reference model for requirements traceability [11], Vilela et al.'s safety requirements metamodel [14], and Goal Structuring Notation standard [15]. The semantics of concepts and relationships are described in the supplementary material in [13].

When teams manage assurance rationale information only inside assurance case reports, the copies and references to project information are duplicated, leading to much effort to keep them up to date throughout the development lifecycle. Using our approach, assurance rationale information management can follow the same workflow as other project items.

The incorporation of more data into project management tools brings new possibilities for dealing with traceability needs. Using our information model as the underlying traceability information model, a team is able to manage assurance information together with the regular data stored in the management tool. Thus, the parsing and analysis of data for developing certification documentation benefit from a richer set of elements and relationships. For automated information retrieval approaches, an enriched tracking system provides valuable data to generate traceability matrices and more custom fields for natural language processing.

1) Mapping the information model: A tracking system can be customized to map the concepts and relationships of our information model. This illustrative scenario shows how to provide support to our information model using Atlassian Jira. Jira provides some predefined issue types to handle software projects and also allows the creation of custom issue types and custom fields. Table I depicts some examples of how Jira was customized to map our information model concepts and relationships, using built-in features and new custom types of issues, types of issue links, and fields. Using the customized Jira, we can record assurance rationale and regular project information together. The adaptation comprises new issue types, additional custom fields, and new issue link types. Once this adaptation is in place, it is possible to record

the safety assurance information, thus enriching the existing project information.

B. Safety Assurance Ontology

We developed an ontology to support representing, assessing and classifying assurance and regular project information. The ontology builds on the information model and incorporates constraints, rules of formations, and competency questions. The main purpose is to support teams to continuously evaluate the quality of project information and its traceability.

The ontology handles the project information at the granularity of single, uniquely identified items such as an issue in a tracking system. 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.

The purpose of ontology is to integrate common and interrelated knowledge of two systems engineering practices: safety engineering and software engineering. The information model presented in section IV-A is the conceptual model of our ontology. The information model was co-developed with 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.

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 [16]. The questions should address the informational needs of team members reviewing assurance information or constructing assurance cases. Table II lists some of the competency questions that comprise the specification of the ARCADE ontology. This set of questions specifies types of items, relationships and attributes that need to be handled. Additionally, we extract rules and constraints from the metamodels in which our information model is based on (see Section IV-A).

We resort to semantic web technologies and choose OWL/DL (Ontology Web Language/Description Logic) as our logic system [17]. The classes, object properties, data properties and rules that comprise the ontology specification can be found in the supplementary material [13]. OWL/DL is sufficiently expressive to represent project data repositories, and the core reasoning is decidable (i.e., the reasoner will not run forever). Also, the use of OWL/DL brings some capabilities for inferring implicit classification based on inward and outward relationships, based on chain of relationships and based on attributes. Finally, complex rules/constraints can be implemented using SWRL, and ontology data can be queried using SPARQL [17]. This technology stack supports many ways to extend the framework to specific teams needs.

This ontology is classified as a *Task-Specific Ontology* because it embodies just the conceptualizations that are needed for carrying out a particular task [18].

TABLE I
INFORMATION MODEL MAPPING INTO JIRA.

Model concept	Representation in Jira
Assumption	**Assumption** field (custom)
Cause	**Cause** issue (custom)
Component	**Component** configuration
Context	**Context** issue (custom)
Design Def.	**Design Definition** issue (custom)
Hazard	**Hazard** issue (custom)
Justification	**Justification** field (custom)
Object	Any kind of issue, labels and components
Requirement	**Story** issue
Safety Req.	**Story** issue with **Label** "Safety"
Source	Any **weblink/attachment** associated to an issue
Strategy	**Strategy** issue (custom)
Model relation	**Representation in Jira**
causes	***causes*** issue link (custom)
contextualizes	**Context** issue linked by ***relates to*** to any issue
documents	**Weblinks/attachments** associated with an issue
explains	***explains*** issue link (custom)
refines	***refines*** issue link (custom)
mitigates	***mitigates*** issue link (custom)
realizes	**Component** associated to **Story** issue
implements	**Component** associated to **Design Definition** issue

TABLE II
COMPETENCY QUESTIONS

Basic
Which are the Objects and their classifications?
Which are the relationships associated with an Object?
Which are the Hazards?
Which are the Safety Requirements?
Which are the Requirements?
Which are the Design Definitions?
Which are the Causes?
Which are the Contexts?
Which are the Components?
Which are the Assumptions?
Which are the Justifications?
Which are the Strategies?
Which are the Sources?
Traceability gaps ("Which are the...")
... Requirements that are not allocated in any Component?
... Safety Requirements that do not mitigate any Hazard?
... Safety Requirements that are not explained by any Rationale?
... Hazards that are not mitigated by any Safety Requirement?
... Causes that are not cause of any Hazard?
... Hazards that have no Cause associated?
... Hazards that are not explained by any Rationale?
... Hazards that are not contextualized by any Context?
... Rationales that do not explain any Object?
... Rationales that are not classified as any of the subclasses?
... Contexts that do not contextualize any Object?
... Components that are not allocated to any Requirement?

C. ARCADE Engine

ARCADE Engine is a web system that orchestrates various third-party libraries for automating the loading, reasoning, querying, and presentation of the ontology data, and also the generation of assurance cases.

The engine is preset with the ontology specification, and the main flow starts upon the input of project data using the interchange format. In the field of ontology, the specification is called TBox (terminology), i.e., the terms that define the concepts, relationships, and rules of the domain being modeled. The input data is called ABox (assertions), i.e., the facts known about the individuals in the domain. Therefore, the project data represent the facts about the system that should be verified against the terms defined.

1) Project Information Retriever: The task of the retriever is to transform the project data extracted from the management tool into semantic triples. A semantic triple (or RDF triple) is the atomic data entity that enables any knowledge to be represented in a machine-readable way [17]. An important feature of RDF is that almost any knowledge repository may be represented as a collection of triples (a RDF graph) [17]. By using the semantics of our ontology and the RDF syntax, we provided a canonical interchange format to receive data from different tools.

A new retriever shall be developed towards each project management tool that is integrated into the framework (see Fig. 1). The information retrieved by multiple retrievers may be combined in the main engine and processed together. Even without a custom retriever implementation, project data can be extracted from a management tool and mapped into the interchange format if teams wish to perform a one-off analysis of their data.

2) Consistency analysis: The main objective of an ontology is to give users the answers to the competency questions. However, before delving into the questions, users need to know if there are any inconsistencies regarding the ontology. The semantic reasoner evaluates the input data (ABox) against the constraints and rules defined (TBox). If the engine finds inconsistencies, it provides an explanation indicating the set of data entries and axioms that lead to the inconsistency. It is up to users to devise which data entry is incorrect, or if some information is missing.

Only viewing the data entries involved in the inconsistency may not provide enough advice for users. So, the engine presents the explanations showing the data entries and axioms involved separately. The axioms are termed *Rules* and data entries as *Project items* in the GUI to ease understanding by users (see Fig. 3).

How to fix an inconsistency? It is expected that team members using the ARCADE web tool have some level of access to the project management tool. So they can either fix the inconsistencies directly in the tool, or notify those responsible for. ARCADE does not act back on the management tool to update project items (issues, fields or issue links).

3) Knowledge exploration: After the consistency analysis, the engine executes queries to gather answers for the competency questions. Some questions are just to find items of specific classes, and other questions aim to find inconsistencies that we decided not to be computed by the reasoner. The motive is that these are not strong flaws in the data and may be related to the temporary incompleteness that the project experiences in the early stages of development.

ARCADE web tool shows the answers to the competency questions in two perspectives: project items and links listing, and direct answers to questions. ARCADE lists all the data retrieved from the project management tool as *provided*. Then the engine runs the reasoning and classification, and all new data implied by the ontology is referred to as *inferred*. Fig. 4 shows the presentation of project items and links. This listing provides the answers for the first two questions (see Table II), thus providing an overview of the entire knowledge database. The list is sorted by item name, presenting the item attributes (data properties). In the rest of the tool, every mention to items is linked back to this listing for easy navigation.

For the other questions, we present the answers, as shown in Fig. 5. We present the question, the answers (if any) found in the dataset, and an indication if the answer is according to the expected. *Basic* questions are expected to be non-empty. *Traceability Gap* questions are expected to be empty, i.e., there should be no project items that fit the query.

4) Assurance Case Generator: Developers face difficulty selecting the proper project data and structuring the argumentation [1]. On the other hand, there are a variety of patterns to organize the argumentation within an assurance case [19]. Using the classified ontology computed by the engine, the generator creates multiple assurance case fragments based on patterns available in the literature. These assurance case fragments provide users with multiple argumentation perspectives based on the same project data.

Each pattern is implemented as a heuristic that queries the ontology model to build the fragment in GSN notation [15], a popular argumentation notation used in academy and industry [2], [7]. To ease extending the framework with new patterns or argument notations, we used the Builder design pattern. Whenever the engine processes new project data, it generates assurance case fragments based on loaded data and selected argumentation patterns. Figure 6 shows part of the structured argument generated based on the pattern *Functional Decomposition* [19]. It breaks down the project data starting from high-level requirements (functionality) that need to be satisfied down to refinements and evidence.

#	Items/Links involved	Rules impacted

Expl. #1	`GPCA-1 is a Requirement`
	`GPCA-10 refines GPCA-1`
	`GPCA-10 is a DesignDefinition`
		– Everything that refines or is refined by a Requirement, it's also a Requirement.
		– Project items can be classified to only one of these classes: Component, Context, DesignDefinition, Hazard, Rationale or Requirement (mutually exclusive).

Fig. 3. Inconsistency explanation

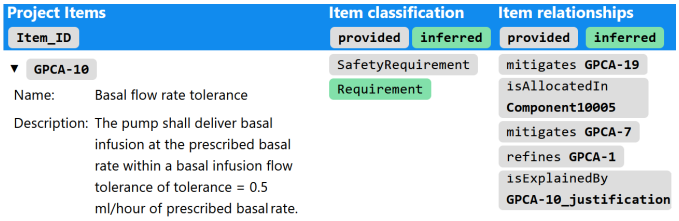

Fig. 4. Project data provided and inferred

Traceability Gap Questions	Status	Items founds
CQ13. Which are the Requirements that are not allocated in any Component	Gaps found!	GPCA-1 GPCA-2
CQ14. Which are the Safety Requirements that do not mitigate any Hazard	No gaps found.	

Fig. 5. Answers for competency questions

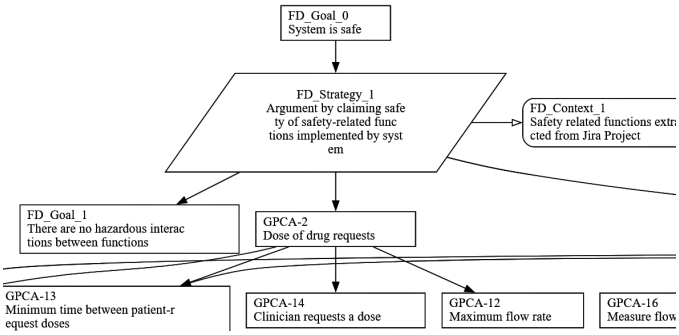

Fig. 6. Assurance case fragment for Functional Decomposition pattern

V. Conclusions

Practitioners developing safety-critical system and assurance cases lack proper tools to manage and review assurance information, and to map them to assurance case arguments. We proposed the ARCADE framework to support teams with a collaborative place where the interrelations between assurance and regular project information can be explicitly managed together early and continuous throughout the system development lifecycle. It leverages early and continuously quality assessment of assurance information recorded in the project management tools, as well as the automated generation of assurance cases.

The envisioned benefits of ARCADE are: **(B1)** Improve team collaboration towards safety assurance; **(B2)** Reduce effort to retrieve and assess assurance information; **(B3)** Reduce effort to build assurance cases; **(B4)** Increase argumentation quality.

ARCADE supports new or improved team activities that leverage these benefits, as detailed in the following.

B1. Instead of managing safety assurance information in separate documents or tools, we support integrated management with project management tools. A unified place for storing, retrieving, and cross-referencing eases traceability management, thus facilitating cross-collaboration across requirements engineering and other activities through the development lifecycle.

B2. Our framework provides automated extraction and analysis, helping users to review consistency and completeness of the project data. Also, by means of the interchange format for data input, the framework can work with data retrieved from different management tools.

B3 and B4. The maintenance of standalone assurance cases may be cumbersome as the project evolves. We argue that there is no strong need to build complete assurance cases during most of the project lifecycle, and preliminary and interim assurance cases [2] could be generated automatically from data available in management tools. Thus, ARCADE provides automated generation of assurance case fragments in multiple perspectives, which can be useful for safety engineers to assess the quality of evidence throughout the development process.

Future works include evaluating the use of ARCADE in industry settings, to assess its adaptability and efficacy to different regulations, projects and teams.

References

[1] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," *Safety Science*, vol. 59, pp. 55–71, 2013.

[2] T. Kelly, "Safety cases," in *Handbook of Safety Principles*, N. Möller, S. Hansson, J. Holmberg, and C. Rollenhagen, Eds. Wiley, 2018, pp. 361–385.

[3] "ISO/IEC/IEEE 15026-2 – systems and software engineering – systems and software assurance – part 2: Assurance case," Standard, Nov. 2022.

[4] T. Myklebust, G. K. Hanssen, and N. Lyngby, "A survey of the software and safety case development practice in the railway signalling sector," in *European Safety and Reliability Conf. (ESREL)*, 2017.

[5] J. Cheng, R. Metoyer, J. Cleland-huang, N. Dame, and N. Dame, "How Do Practitioners Perceive Assurance Cases in Safety-Critical Software Systems ?" in *Proc. of the 11th Intl. Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, 2018, pp. 5–8.

[6] L. E. G. Martins and T. Gorschek, "Requirements engineering for safety-critical systems: A systematic literature review," *Information and Software Technology*, vol. 75, pp. 71–89, 2016.

[7] C. Almendra, C. Silva, L. E. G. Martins, and J. Marques, "How assurance case development and requirements engineering interplay: A study with practitioners," *Requirements Engineering*, vol. 27, no. 2, p. 273–292, jun 2022.

[8] "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, Apr. 2021.

[9] C. Almendra, C. Silva, and J. Vilela, "Incremental development of safety cases: a mapping study," in *Proc. of the 34th Brazilian Symposium on Software Engineering*, 2020, pp. 538–547.

[10] C. Almendra and C. Silva, "Managing assurance information: A solution based on issue tracking systems," in *Proc. of the 34th Brazilian Symposium on Software Engineering*, 2020, pp. 580–585.

[11] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, Jan 2001.

[12] P. Mäder, P. L. Jones, Y. Zhang, and J. Cleland-Huang, "Strategic traceability for safety-critical projects," *IEEE Software*, vol. 30, no. 3, pp. 58–66, May 2013.

[13] C. Almendra and C. Silva, "Supplementary material for "ARCADE: a Framework for Integrated Management of Safety Assurance Information"," UFC, UFPE, Tech. Rep., 2023. [Online]. Available: https://arcade-framework.github.io/arcade-renext/

[14] J. Vilela, J. Castro, L. Martins, and T. Gorschek, "Safe-RE: a Safety Requirements Metamodel Based on Industry Safety Standards," in *Proc. of the 32nd Brazilian Symp. on Softw. Eng.*, 2018, pp. 196–201.
[15] "Goal structuring notation community standard (version 3)," The Assurance Case Working Group, Standard, May 2021.
[16] M. C. Suárez-Figueroa, A. Gómez-Pérez, and B. Villazón-Terrazas, *How to Write and Use the Ontology Requirements Specification Document*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 966–982.
[17] World Wide Web Consortium. Semantic web standards. [Online]. Available: https://www.w3.org/2001/sw/wiki/Main_Page
[18] G. Schreiber, "Knowledge engineering," *Foundations of Artificial Intelligence*, vol. 3, pp. 929–946, 2008.
[19] M. Szczygielska and A. Jarzębowicz, "Assurance case patterns on-line catalogue," in *Advances in Dependability Engineering of Complex Systems*. Cham: Springer International Publishing, 2018, pp. 407–417.