Pioneering the Creation of ISO 26262-compliant OSLC-based Safety Cases

Barbara Gallina
Mälardalen University
Västerås, Sweden
barbara.gallina@mdh.se

Mattias Nyberg
Scania AB
Södertälje, Sweden
mattias.nyberg@scania.com

***Abstract*—ISO 26262 requires for each item the creation of a safety case. Such creation is extremely time-consuming. Currently, no satisfying approach is at disposal to speed up such creation. OSLC (Open Services for Lifecycle Collaboration) is a standard for tool interoperability, which, if enabled, permits effective documentation management, needed for efficient safety case creation. OSLC defines a set of extensible core specifications (domains), each of which focuses on a single phase of the life-cycle. In our previous work, we provided ISO 26262-compliant domain extensions. In this paper, we use such extensions to pioneer the creation of OSLC-based safety cases. In particular, we show how information exposed via such extensions can be queried to "produce" compositional pieces of safety case-fragments, arguing about requirements traceability and satisfiability. We illustrate the production of such fragments for an Electronic Control Unit-module in use at Scania. We then discuss our findings.**

Keywords—ISO 26262; safety cases; OSLC; SPARQL Protocol and RDF Query Language (SPARQL).

I. INTRODUCTION

ISO 26262 [1] is the standard for automotive functional safety. Initially introduced for road vehicles up to 3,5-ton gross mass, ISO 26262 is now under revision to be proposed for all road vehicles, including heavy trucks. A new version of the standard is expected to be issued by 2018. ISO 26262 requires for each item the creation of a safety case. According to ISO 26262- Part 1, Definition 1.106, "a safety case is an argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development". To show requirements satisfiability, a safety case is expected to compile all the work products of the life-cycle in a traceable manner. Thus, an effective documentation management is necessary.

The creation of a safety case is extremely time-consuming, involving hundreds of work-products. Its manual creation does not seem to be an option due to its complexity, which is even higher in the case of trucks. Keeping safety cases up-to-date fully manually would subtract precious time to the engineering phase. These observations stem from our own experience. Since the introduction of ISO 26262, we have been working on methods for building safety cases in compliance with ISO 26262 (see [4] and [5]). Initially, we have targeted manual creation and then we have started conceiving semi-automatic creation by proposing model-driven certification approaches. Concretely, we have shown that a safety case fragment can be created semi-automatically via transformation rules from contract-and component-based architectural specifications [13] and process models [11]. Moreover, we have proposed a Cloud-based infrastructure (see [12]), where safety processes, including tasks aimed at generating safety case fragments can be enacted on the Cloud. In the literature, other model-based approaches have been explored for enabling the semi-automatic generation of safety case fragments. Currently, however, no satisfying and fully integrated approach exists for enabling semi-automatic creation of skeletons of safety cases, embracing the tool-supported portion of the safety life-cycle. Compositional approaches are envisioned however their implementation is hindered by the limited tool interoperability.

OSLC (Open Services for Lifecycle Collaboration) [31] is a recently introduced standard, built on top of the Semantic Web standards, aimed at enabling life cycle tools interoperability and effective documentation management via production and consumption of resources. As presented in our previous work [8], an ISO 26262-compliant tool chain can be achieved by extending such domains. Such a tool chain would rely on a ISO 26262-compliant ontology, offering a unified representation, which may facilitate compositional creation of safety cases. In this paper, based on what previously envisioned [6], we pioneer the creation of OSLC-based safety cases, towards semi-automatic and continuous self-assessment. More specifically, we first interconnect our previously defined ISO 26262-compliant OSLC domains. Then, we show how information exposed via such domains can be queried in order to first get essential evidence and reasoning via a set of SPARQL queries and then use such evidence and reasoning to generate compositional pieces of safety case-fragments for arguing about requirements traceability and satisfiability. Finally, we illustrate the manual generation of such fragments for a Scania Electronic Control Unit-module and we discuss our findings.

The rest of the paper is organized as follows. In Section II, we provide background information. In Section III, we provide an overview of the entire approach. In Section IV, we identify the resources that are needed for the safety case compilation and we represent them as ontologies. In Section V, we query the ontological representation to check for traceability and supportive evidence. In Section VI, we create safety case fragments. In Section VII, we discuss our findings. In Section VIII, we discuss related work. Finally, in Section IX, we present our concluding remarks and future work.

II. BACKGROUND

In this section, we present the background information on which we base our work.

A. ISO 26262-compliant Software Unit Design and Testing

In this subsection, we limit our attention to a subset of clauses (8-9) of Part 6. These clauses target the software unit's development within the software V-model. More specifically, clause 8 defines how a software unit should be designed and implemented; while clause 9 defines how a software unit should be tested against its design specification. Expected work products related to clause 8 are: Software unit design specification and Software unit implementation. Expected work products of clause 9 are: Software verification plan, Software verification specification, and Software verification report. These work products should be produced in compliance with the process-related requirements stated in the clauses 8-9 plus all other related requirements stated in other clauses. These requirements define the properties that the work products should exhibit, the methods to produce them, etc.

B. CMS (Chassis Management System)1

CMS1 is an ECU (Electronic Control Unit), which contributes to the realization of the *Fuel Level Estimation and Display System* within Scania products. CMS1 is responsible for calculating the total fuel level. Within CMS1 there is a software module, called *CMS1: Fuel*. *CMS1: Fuel*'s design, implementation and testing is conducted according to practices adopted for a large number of ECUs developed at Scania. Thus, CMS1: Fuel can be considered a representative case.

C. OSLC, RDF(S), and SPARQL

OSLC [31] is a standard that defines a set of core specifications to enable interoperability of tools, used during a product's life cycle. The set of specifications, called domains, target e.g., requirements management (RM), Architecture Management (AM), and Quality Management (QM). OSLC builds on top of Linked Data [24], Resource Description Framework (RDF) [16], RDF Schema [17], and HTTP protocol. Each work product is described as an HTTP resource, identified via a Uniform Resource Identifier (URI). To interoperate via a work product, a tool that acts as a provider has to associate an URI to the work product and post it; a tool acting as consumer can get the work product from the URI itself.

RDF [25-26] provides a standard representation for data as directed graphs to facilitate the linking of the resources to be described. This standard representation defines a key data structure: the *RDF graph*, also called triple (Subject, Predicate, Object). In RDF terms, the subject is the thing being described (a resource identified by an URI), the predicate is a property type of the resource, and the object is equivalent to the value of the resource property type for the specific subject. The core RDF vocabulary is defined in an XML namespace, commonly called *rdf*, where the URI is http://www.w3.org/1999/02/22-rdf-syntax-ns#. The following commented XML/RDF-based fragment shows an example of an RDF resource, where a software unit resource is described.

```
<?xml version="1.0"?>
<!—Comment: rdf document with prefix rdf and ex (selected namespace) -->
<rdf:RDF
xmlns:rdf="http://www.w3. org/1999/02/22 rdf syntax ns#"
xmlns:ex="http://example.com/elements/1.0/">
<!-- Definition of the resource software unit -->
    <ex:SoftwareUnit>
    <!--property identifier -- >
    <ex:documentedIn>A02</ex:documentedIn>
     <!--property name-->
    <ex:name>CMS1:Fuel </ex:name>
    </ex:SoftwareUnit>
</rdf:RDF>
```

RDF Schema (RDFS) [17] provides a data-modelling vocabulary for RDF data. RDFS permits groups of related resources and the relationships between these resources to be specified.

SPARQL [15, 23] is a recursive name and stands for SPARQL Protocol and RDF Query Language. SPARQL can be used to express queries across diverse data sources (which can be stored natively as RDF graphs). As summarized in [23], SPARQL specifies four different query variations for different purposes: 1) SELECT query is used to extract raw values, the results are returned in a table format; 2) CONSTRUCT query is used to extract information and transform the results into valid RDF; 3) ASK query is used to provide a simple True/False result for a query. Finally, 4) DESCRIBE query is used to extract an RDF graph. In our work, SELECT, CONSTRUCT and ASK queries are the most used.

D. Safety Case Documentation

As extensively discussed in [22], several documenting approaches (textual and/or graphical) exist to structure a safety case. The most common graphical and human-readable approaches have been unified and standardized via the OMG standard SACM (Structured Assurance Case Meta-model) [18], which provides language constructs to model the argumentation aimed at explaining why the claims are (not) supported as well as the (counter) evidence aimed at providing the foundation for the (counter) claims.

III. APPROACH OVERVIEW

In this section, we provide an overview of our approach, which enables continuous self-assessment by seamlessly aligning the life-cycle of a safety case with the life-cycle of the product. Thus, self-assessment can continuously semi-automatically be performed by compiling the different types of evidence. Fig. 1 depicts the overview of our approach. More specifically, Fig. 1 limits its focus to the alignment of a portion of the software V-model and the compilation of the evidence related to that portion. A user (in this case, the safety architect, software component developer, and the safety manager) may preliminarily execute SPARQL queries of type "ASK" to make certain that the rational (reasoning steps) have been documented, i.e., that the traces linking pieces of evidence and (sub)claims exist and that explanations are offered where necessary. If this is the case, then, the safety case generator, embraced by the dotted-line in Fig.1, can execute SPARQL query of type "CONSTRUCT" in order to create and populate argumentation-related RDF-graphs, which are expected to be compliant to a specific argumentation meta-model. Finally, this model can be queried (via SPARQL query of type "SELECT") and via a model-transformation the retrieved information, can be used to build e.g. GSN-goal structures in compliance with SACM.

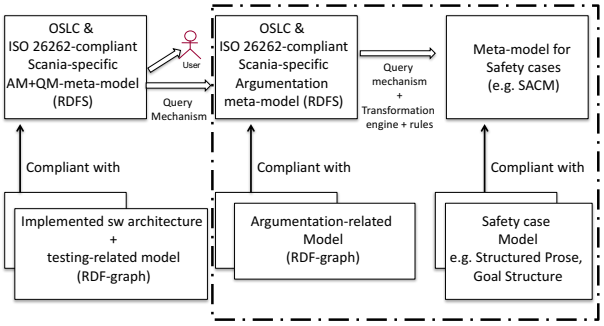

Fig. 1. OSLC-based approach for self-assessment.

IV. ISO 26262-COMPLIANT AM & QM INSTANCES

The software V-model described within ISO 26262-Part 6, can be sliced and mapped onto three OSLC-based domain extensions: one aimed at representing the requirements engineering phase (ISO 26262-compliant OSLC RM), one aimed at representing the design and implementation phase (ISO 26262-compliant OSLC AM) and finally one aimed at representing the verification phase (ISO 26262-compliant OSLC QM). In the context of two master theses (see [10] and [9]), methodological guidelines for designing ISO 26262 – compliant AM and QM were provided. Additionally, RDFS representations of the AM and QM domains were separately provided and RDF-graphs were created based on Scania documentation and only partly published (see [7] and [8]).

In this section, we limit our attention to a very limited portion of the AM and QM-related extensions. More specifically we focus on few classes (namely, SW Unit Implementation, SW Verification Report, SW Verification Specification, and, Object to be tested) and create instances by populating them with *CMS1: Fuel* –related information and by linking them. The following listings partially represent the instances.

<!--**SW Unit Implementation**: CMS1: Fuel -->
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:oslc="http://open-services.net/ns/core#"
xmlns:oslc_iso26262am="http://open-services.net/ns/oslc_iso26262am#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<oslc_iso26262am:SWUnitImplementation
rdf:about= " http://open-services.net/ns/oslc_iso26262am/SWUnitImplementation/CMS1Fuel">
…
</oslc_iso26262am:SoftwareUnitImplementation>
</rdf:RDF>

<!—**SW Verification Report**-->
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:oslc_iso26262qm="http://open-services.net/ns/iso26262qm#">
<oslc_iso26262qm:SWVerificationReport
rdf:about=" http://open-services.net/ns/oslc_iso26262qm /verificationReports/1">
<dcterms:description> Work product, specified according to ISO 26262-Part6, 9.5.3, that consists of the execution and evaluation of the software with reference to the software verification plan and software verification specification</dcterms:description>
<dcterms:identifier> 1 </dcterms:identifier>
<dcterms:title >SW Verification report </dcterms:title >
<oslc_iso26262qm:**passResult**>1 </oslc_iso26262qm:passResult >
…
<oslc_iso26262qm:usesSWVerificationSpecification rdf:resource="
http://open-services.net/ns/oslc_iso26262qm/verificationSpecifications/1" />
</oslc_iso26262qm:SWVerificationReport>
</rdf:RDF>

<!—**SW Verification Specification**-->
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:oslc_iso26262qm="http://open-services.net/ns/iso26262qm#">
<oslc_iso26262qm:SWVerificationSpecification
rdf:about=" http://open-services.net/ns/oslc_iso26262qm/SWVerificationSpecification/1">
<dcterms:description> Specification of methods, test environment, execution of CMS1:Fuel. It includes the resources of test objects and test cases used to test these objects </dcterms:description>
<dcterms:identifier> 1 </dcterms:identifier>
<oslc_iso26262qm:testlevel> sw unit test</oslc_iso26262qm:testlevel>
<oslc_iso26262qm:**objectToBeTested**
"http://open-services.net/ns/oslc_iso26262am/SWUnitImplementation/CMS1Fuel">
…
</oslc_iso26262qm:SWVerificationSpecification>
</rdf:RDF>

<!—**Object to be tested: CMS1:Fuel** -->
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms=http://purl.org/dc/terms/
xmlns:oslc_iso26262qm="http://open-services.net/ns/iso26262qm#">
<oslc_iso26262qm:ObjectToBeTested
rdf:about=
"http://openservices.net/ns/oslc_iso26262am/SWUnitImplementation/CMS1Fuel **">**
<dcterms:description> CMS1:Fuel description</dcterms:description>
<dcterms:identifier> 1 </dcterms:identifier>
<dcterms:title> CMS1:Fuel </dcterms:title>
<oslc_iso26262qm:testlevel> sw unit test</oslc_iso26262qm:testlevel>
<oslc_iso26262qm:objectType> sw unit </oslc_iso26262qm:objectType>
<oslc_iso26262qm:usedBySoftwareTestCase rdf:resource=" http://open-services.net/ns/oslc_iso26262qm /testCases/1" />
<oslc_iso26262qm:usedBySWVerificationSpecification rdf:resource="
http://open-services.net/ns/oslc_iso26262qm/verificationSpecifications/1" />
</oslc_iso26262qm:ObjectToBeTested >
</rdf:RDF>

For sake of clarity it should be noted that for confidentiality reasons, these instances are based on Scania documents related to an old variant of the CMS1: Fuel. As it can be easily retrieved from the listing, a software verification report is connected to the software verification specification via “uses”. The software verification specification contains the object to be tested, which points to the specific software unit implementation (CSM1: Fuel). The explanation of all the properties of the above-listed resources is out of scope. The interested reader might refer to ISO 26262-Part 6.

V. Querying the Ontological Representation

In this section, we first present the infrastructural settings for querying the ontological representation. Then, we explain which kind of queries we intend to execute. Finally, we provide an example of such queries.

In our environment, the user performs queries via a SPARQL endpoint and gets the needed results, based on data stored in graph DB which is fed with information coming from the tools used for software unit requirements specification, design, implementation, and testing. Implemented SW Units – IDE contain the work products related to clause 8 while the testing tool contains information related to the work product required by clause 9.

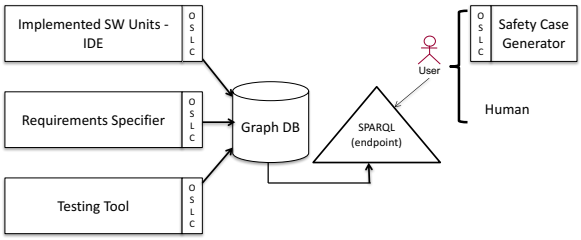

Fig. 2. Infrastructural settings

As Fig. 2 shows, the user might be a human being (e.g., safety engineer) or a machine, more specifically a safety case generator that coherently with what depicted in Fig. 1 would get results to be then visualized in a standardized format.

Since the safety case generator has not been developed yet, in this section, we only provide initial sets of conceptual queries to be formulated in SPARQL by a human. It should be noted that, given our approach, three different kinds of queries can be envisaged. The first set of queries of type CONSTRUCT is aimed at constructing argumentation-related RDF graph from RDF graphs related to ISO 26262-Part 6-resources in order to populate the RDF-graph representing the argumentation. CONSTRUCT queries should enable the complete compilation of ISO 26262 Part 6 work-products.

The second set of queries of type SELECT is aimed at retrieving the information to be used by transformation rules to render the argumentation via the popular concrete syntaxes [22]. For instance, the leader of the testing-team might be interested in inspecting the argument concerning traceability and satisfiability of software requirements and see if the software unit implementation has been tested and if the result was "pass" or not. The identification and visualization of counter evidence is crucial and should trigger a re-implementation of even a re-design.

Finally, the third type of queries of queries of type ASK is aimed at asking questions to get quick confirmations. For instance, an external assessor (e.g., an auditor) might be interested in checking if the software unit design is designed according to the method appropriate for its criticality level [9]. An external assessor might also be interested in checking if, for a given software verification report, the corresponding software verification specification exists.

In this paper, we only present one simple ASK query:

```
PREFIX oslc_iso26262am: <http://open-services.net/ns/oslc_iso26262am#>
PREFIX oslc_iso26262qm: <http://open-services.net/ns/oslc_iso26262qm#>
ASK{
 { ?subject oslc_iso26262qm:passResult ?o
   FILTER(xsd:integer(?o="1"))}
}
```

This query was performed on the graph obtained by considering jointly the AM and the QM domains. This query returned YES since the verification report under consideration contains a single test case with passResult equal to one. This query was executed by using TopBraid Composer, the namespaces where not published.

VI. Creating a Safety Case Fragment

In this section, we sketch a pattern-based safety case fragment that can be used to construct an instance via our approach based on the current information presented within the ISO 26262-compliant AM and QM. To do that we use simple declarative language, indentation, numbering, etc. as proposed by [22].

Claim 1: Algorithm X was successfully tested.
Context 1: Definition of successfully tested via coverage criteria. Definition of X.
Claim 1.1: All critical test cases passed
Context 2: Definition of critical test cases.
Strategy 1.1: Argument over all critical test cases (TC1, TC2, TCN)
Claim 1.1.1: Test case TC1 passed
Evidence 1.1.1: Test report to be directly linked to TC1;
Claim 1.1.2: Test case TC1 passed
Evidence 1.1.2: Test report to be directly linked to TC2;
...
Claim 1.1.N: Test case TC1 passed
Evidence 1.1.N: Test report to be directly linked to TCN;

By replacing X with "CMS1:Fuel" and by considering that in our simple example only 1 test case was considered, we obtain:

Claim 1: CMS1:Fuel was successfully tested.
Context 1: Definition of successfully tested via coverage criteria.
Claim 1.1: All critical test cases passed
Context 2: Definition of critical test cases.
Strategy 1.1: Argument over test case TC1
*Claim 1.1.1: Test case TC1 ("*http://open-services.net/ns/oslc_iso26262qm*/testCases/1") passed*
Evidence 1.1.1: Test Execution Log (rdf:resource= http://open-services.net/ns/oslc_iso26262qm/*testExecutionLogs/1);*

This fragment is not intended to be comprehensive. The degree of coverage with regard to what is expected to be presented in a safety case is extremely limited. This fragment is not intended to be compelling either. Additional years of experience are required to provide compelling safety cases for truck-related items. As mentioned in the introduction, a new version of the standard embracing all road vehicles is expected

to be issued by 2018. Thus, a comprehensive and compelling argument is still to be developed.

Similar fragments could be conceived for showing process compliance. However, at the time being no process compliance could be claimed since ASIL-classification was not yet integrated within the documents that we considered for this paper. By formulating adequate queries, counter evidence could be identified and this would be beneficial since it could lead to mitigation actions aimed at increasing safety.

VII. Discussion

Semi-automatic argument generation of argument fragments might be considered inappropriate. The risk could be that only supportive evidence is considered. To avoid being biased by the well-known confirmation bias, in our approach queries aimed at identifying counter evidence are also expected to be formulated. Moreover, as mentioned our intention is to offer an approach for continuous self-assessment. The identification of counter evidence is expected to trigger a review/redo of previous process steps.

The benefits of using OSLC to enable traceability is undoubtable. Our vision was to bring those benefits to safety-critical systems self-assessment. Our vision-oriented and breadth-first-oriented investigation is still in its early stages and we have not yet performed a proper evaluation of our approach. Evaluating our approach is indeed challenging as the resources (time and workforce) are not available to develop the OSLC adaptors as well as other tools -required to create a complex, real world safety case using our approach. Therefore, in our work we rely on a phased evaluation in which we use the lessons learnt from our experiences with ISO 26262-Part 6, OSLC-domain extension, and Apache Jena [29] in addition to studying literature and learning from industrial experience as in [30] to validate the potential of our approach.

In our pioneering and conceptual work, no issue concerning e.g., maturity of OSLC, scalability when performing complex queries, was taken into consideration. As surveyed in [28], OSLC is still unstable to offer a solution spanning the entire ALM-tool chain. However, given its potential, we believe that it is worth investigating this technological domain and in parallel contribute to its development. Given our initial simple queries and the current infrastructural settings, where data can be considered static, we selected SPARQL. However, to perform continuous self-assessment in the presence of a real-time stream of data, other query languages could be explored. For instance, Continuous SPARQL [32], the extension of SPARQL to query RDF streams could be taken into consideration.

VIII. Related work

In this section, we discuss work that is related to ours either because of similar choices in terms of OSLC-specifications or because of similar objectives in terms of querying mechanisms. Moreover, we also discuss work aimed at extracting automatically arguments from unstructured textual corpora.

In the literature, few works have currently explored the semi-automatic creation of safety cases based on OSLC. Iliasov et al 2015 [20] present their vision for building an OSLC-based prototype of integrated environment for engineering and certifying dependable systems. Laibinis et al 2015 [21] further develop the work presented in [20]. Concerning transformation rules, declarative transformation rules from RDF to RDF and other languages were discussed in [14]. Authors also discussed how to make their approach generic, i.e., the rule language independent from the output language.

Concerning querying mechanisms, Denney et al. 2014 [19] introduce a preliminary approach and a new query language called AQL (Argumentation Query Language). Via their approach, they semantically enrich GSN arguments with domain-specific metadata that the query language leverages to produce views and offer a means to get answers to specific questions. In our approach, a new language is not needed. The exploitation of the semantic web already offers semantics-enriched data, which can be queried to obtain desired information. Finally, concerning automatically extraction of arguments from unstructured textual corpora, recently, an online argumentation mining system, called MARGOT, was proposed [28]. Via MARGOT, claims and evidence detection with word-level granularity is supported. However, MARGOT has not yet been applied to safety-critical systems.

IX. Conclusion and future work

In this paper, we have built on top of our previous work and we have performed an additional step towards an ISO 26262-compliant OSLC-based tool chain enabling continuous self-assessment, via the continuous semi-automatic creation of safety cases. Our step consisted of the provision of a global vision of such a tool chain and a set of SPARQL queries, aimed at extracting information from an RDF-graph, representing an instance of interconnected domains targeting ISO 26262-Part 6. As discussed, in principle, our intended set of queries supports the compilation of traceable work product towards the semi-automatic creation of a safety case. To empirically investigate the effectiveness of our domain, we have instantiated it for a truck ECU system at Scania. An RDF-graph embracing the software design, implementation, testing-related resources was created. Finally, a discussion was provided. The approach has been presented for ISO 26262-Part 6. However, it can be extended to the entire ISO 26262, as soon as appropriate domains are available.

In the near future, we plan to develop our approach further by: creating the Scania-specific argumentation meta-model, developing CONSTRUCT queries to populate its models, and more in general by defining a set of queries according to the stakeholder (assessor, safety manager, developer, product manager). In the long-term future, we aim at achieving a tool-supported proof of concept aimed at demonstrating the benefits of having an ISO 26262-compliant OSLC-based tool chain enabling continuous self-assessment, initially limited to ISO 26262-Part 6. To do that, we plan to develop OSLC-adaptors for the Scania -tools related to ISO 26262-Part 6 as well as adapt and integrate the Safety Case Generator, which was initially developed to be executed on the Cloud [12]). Once the tool-chain is in place, we will also investigate possible limitations due to scalability issues and rapidly changing data when performing complex queries.

ACKNOWLEDGMENT

This work has been financially supported by EU and VINNOVA via the ECSEL Joint Undertaking project AMASS (No 692474) [3] and by the Swedish Foundation for Strategic Research via the Gen&ReuseSafetyCase (No SM14-0013) project [2].

REFERENCES

[1] International Organization for Standardization (ISO). ISO 26262: Road vehicles — Functional safety, 2011.

[2] Gen&ReuseSafetyCases. http://www.es.mdh.se/projects/393-genreusesafetycases.

[3] AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems). http://www.amass-ecsel.eu.

[4] R. Dardar, B. Gallina, A. Johnsen, K. Lundqvist, and M. Nyberg. Industrial experiences of building a safety case in compliance with iso 26262. In *IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 349–354, 2012.

[5] B. Gallina, A. Gallucci, K. Lundqvist, and M. Nyberg. VROOM & cC: a Method to Build Safety Cases for ISO 26262-compliant Product Lines. In *2nd Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems*. Hyper Articles en Ligne (HAL), September 2013.

[6] B. Gallina and M. Nyberg. Reconciling the ISO 26262-compliant and the Agile Documentation Management in the Swedish Context. In *Critical Automotive applications: Robustness & Safety (CARS), Matthieu Roy, Paris, France, HAL*, September 2015.

[7] B. Gallina, J. P. Castellanos Ardila, and M. Nyberg. Towards Shaping ISO 26262-compliant Resources for OSLC-based Safety Case Creation. In *Critical Automotive applications: Robustness & Safety (CARS), Gteborg, Sweden, HAL*, September 2016.

[8] B. Gallina, K. Padira, and M. Nyberg, "Towards an iso 26262-compliant oslc-based tool chain enabling continuous self-assessment," in *10th International Conference on the Quality of Information and Communications Technology- Track: Quality Aspects in Safety Critical Systems (QUATIC), Lisbon, Portugal*, 6-9 September, 2016.

[9] J. P. Castellanos Ardila, "Investigation of an OSLC-domain targeting ISO 26262," Master's thesis, Ma¨lardalen University, School of Innova- tion, Design and Engineering, Va¨stera¨s, Sweden, to appear in 2016.

[10] K. Padira. Investigation of Resources Types for OSLC domains Targeting ISO 26262: Focus on Traceable safety evidence for the Right side of the ISO 26262 Software V-model. Master's thesis, Blekinge Tekniska Hgskola, Karlskrona, Sweden, 2016.

[11] B. Gallina. A Model-driven Safety Certification Method for Process Compliance. 2nd IEEE International Workshop on Assurance Cases for Software-intensive Systems (ASSURE), joint event of ISSRE, Naples, Italy, doi: 10.1109/ISSREW.2014.30, pp. 204-209, November 3-6, 2014.

[12] S. Alajrami, B. Gallina, I. Sljivo, A. Romanovsky, P. Isberg. Towards Cloud-Based Enactment of Safety-Related Processes. Proceedings of the 35th International Conference on Computer Safety, Reliability and Security (SafeComp), Trondheim, Norway, September 20-23, 2016.

[13] I. Sljivo, B. Gallina, J.Carlson, H. Hansson. Generation of Safety Case Argument-Fragments from Safety Contracts. Proceedings of the 33rd International Conference on Computer Safety, Reliability and Security (SAFECOMP), Springer, LNCS 8666, ISBN 978-3-319-10505-5, pp. 170-185, Florence, Italy, September 10-12, 2014.

[14] Olivier Corby, Catherine Faron-Zucker. A Transformation Language for RDF based on SPARQL. van der Aalst, W.; Mylopoulos, J.; Rosemann, M.; Shaw, M.J.; Szyperski, C. Web Information Systems and Technologies, Springer, 2015, Lecture Notes in Business Information Processing, <10.5220/0005450604660476>. <http://www.springer.com/>. <hal-01186048>

[15] "SPARQL 1.1 Query Language." [Online]. Available: https://www.w3.org/TR/ sparql11-query/

[16] RDF Primer. http://www.w3.org/tr/rdf-primer/.

[17] RDF Schema 1.1, W3C Recommendation 25 February 2014. http://www.w3.org/tr/rdf-schema/.

[18] OMG. SACM: Structured Assurance Case Metamodel. Technical report, Version 1.1,. http://www.omg.org/spec/SACM, 2015.

[19] E. Denney, D. Naylor, and G. Pai. Querying Safety Cases. *Proceedings of the 33rd International Conference on Computer Safety, Reliability, and Security - Volume 8666* (SAFECOMP). Springer-Verlag New York, Inc., New York, NY, USA, 294-309, 2014.

[20] A. Iliasov, A. B. Romanovsky, L. Laibinis, E. Troubitsyn¨a. OSLC-based Support for Integrated Development of Dependable Systems. Ilir Gashi; Yann Busnel. 11th European Dependable Computing Conference (EDCC), Sep 2015, Paris, France., Proc. of Fast Abstract. <hal-01226607>

[21] L. Laibinis, E. Troubitsyna, Y. Prokhorova, A. Iliasov, A. Romanovsky, From Requirements Engineering to Safety Assurance: Refinement Approach. First International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, SETTA, Nanjing, LNCS 9409, 201–216, Springer, 2015.

[22] C.M. Holloway. Safety Case Notations: Alternatives for the Non-Graphically Inclined? In C.W. Johnson and P. Casely (eds.), Proc. of the IET 3rd International Conference on System Safety, IET Press, Savoy Place, London, 2008.

[23] B. DuCharme. Learning SPARQL, 2nd Edition, Querying and Updating with SPARQL 1.1. O'Reilly Media, July 2013.

[24] Linked Data, "http://www.w3.org/designissues/linkeddata.html."

[25] S. Powers, *Practical RDF*. OReilly Media, 2003. [Online]. Available: https://www.safaribooksonline.com/library/view/practical-rdf/0596002637/index.html

[26] W3C, "RDF 1.1 Concepts and Abstract Syntax," 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-concepts- 20140225/

[27] AMASS, D5.1 Baseline and Requirements for Seamless Interoperability, http://www.amass-ecsel.eu/content/deliverables, Sept. 2016.

[28] M. Lippi and P. Torroni. MARGOT. Expert Syst. Appl. 65, C (Dec 2016), 292-303. DOI: https://doi.org/10.1016/j.eswa.2016.08.050

[29] Apache Jena https://jena.apache.org

[30] J. El-Khoury, D. Gurdur, F. Loiret, M. Törngren, D. Zhang, M. Nyberg. Modelling Support for a Linked Data Approach to Tool Interoperability. ALLDATA: The Second International Conference on Big Data, Small Data, Linked Data and Open Data, 2016.

[31] Open Services for Lifecycle Collaboration. http://open-services.net/.

[32] D.F. Barbieri, D. Braga, S.Ceri, E. Della Valle, M. Grossniklaus. C-SPARQL: SPARQL for continuous querying. In: WWW 1061–1062, 2009.