

A Model-driven Safety Certification Method for Process Compliance

Barbara Gallina

IDT, Mälardalen University, P.O. Box 883, SE-72123 Västerås, Sweden

barbara.gallina@mdh.se

Abstract—A safety case is a contextualized structured argument constituted of process and product-based sub-arguments to show that a system is acceptably safe. The creation of a safety case is an extremely time-consuming and costly activity needed for certification purposes. To reduce time and cost, reuse as well as automatic generation possibilities represent urgent research directions. In this paper, we focus on safety processes mandated by prescriptive standards and we identify process-related structures from which process-based arguments (those aimed at showing that a required development process has been applied according to the standard) can be generated and more easily reused. Then, we propose a model-driven safety certification method to derive those arguments as goal structures given in Goal Structuring Notation from process models given in compliance with Software Process Engineering Meta-model 2.0. The method is illustrated by generating process-based arguments in the context of ISO 26262.

Keywords—*Safety processes, safety cases, process-based arguments, safety standards, model driven engineering, Software Process Engineering Meta-model (SPEM) 2.0, Structured Assurance Case Metamodel (SACM), Goal Structuring Notation (GSN)*

I. INTRODUCTION

Safety standards (e.g. ISO 26262 [1], etc.) define processes (also known as safety life-cycles) to be adopted during the development of safety-critical systems.

According to these safety standards, safety-critical systems are expected to be capable to mitigate the causes of accidents (i.e. hazards). The top-level safety process activities consist of the definition of the system to be developed, identification and categorization of the hazards and risk assessment procedures. Once hazards are identified, they are categorized by assigning a domain-specific safety level. Within the automotive domain (ISO 26262), for instance, a safety level is called ASIL-Automotive Safety Integrity Level- and can assume one out of five values, ranging from negligible QM and A to D, where D represents a hazard that may lead to catastrophic consequences. Once hazards are categorized, safety requirements aimed at reducing risk are elicited as well as traced throughout the traditional development steps (specification, design, implementation, etc.). In parallel, verification and validation activities are carried out to check that the elicited safety requirements are correctly specified, designed, implemented and deployed.

In general, the process definition within standards exhibits different abstraction levels (i.e. domain-independent vs. domain-dependent). Moreover, the process definition is parametric (i.e. activities to be performed, guidelines to be followed, etc. depend on the required safety level). The prescriptive nature of these processes has to do with the

assumption (not always valid as discussed in [2]) that the adoption of best practices during the systems development is correlated to the achievement of good products. For this reason, in some domains, compliance with those processes is required for certification purposes. As discussed in [3], process compliance is of particular value whenever confidence in product-based arguments (supporting safety claims) is limited.

Concerning process compliance, in ISO 26262 we can read:

- “The organization shall institute, execute and maintain organization-specific rules and processes to comply with the requirements of ISO 26262” (Part 2, 5.4.2.2).
- Organization-specific rules and processes for functional safety is a specific work-product that must be provided (Part 2, 5.5.1).
- “A functional safety audit shall be carried out for items, where the highest ASIL of the item’s safety goals is ASIL (B), C, or D, in accordance with 6.4.7, 6.4.3.5 i) and 6.4.8.2.” (Part 2, 6.4.8.1), where a functional safety audit is a work-product aimed at evaluating the process implementation.
- “The organization may tailor the safety lifecycle” (Part 2, 5.4.5.1) and tailoring rules are then detailed.

Thus, for certification purposes, it is crucial to provide work-products (process-based arguments) aimed at showing that either ISO 26262-compliant process activities have been performed or they have been tailored appropriately according to the tailoring rules provided within ISO 26262. The provision of such arguments is expensive and time-consuming since “compiling” the evidence (significant amount of documents) within a compelling argument requires attentive expertise but also unnecessary repetitive work. Currently, no ready-to-use approach exists to reduce time and cost related to the creation of this kind of process-based arguments. Thus, we propose to enable the creation as well as reuse of process-based arguments, by introducing a model-driven safety certification method, called MDSafeCer, that allows safety managers to (semi) automatically generate process-based arguments from process models and avoid unnecessary repetitive work.

The rest of the paper is organized as follows. In Section II, we provide essential background information. In Section III we present the proposed model-driven engineering method aimed at generating process-based arguments from process models. In Section IV, we apply the method. In Section V, we discuss related work. Finally, in Section VI we present some concluding remarks and future work.

II. BACKGROUND







In this section, we present the background information on which we base our work. In particular, in Section II-A, we recall what should be meant by *process* and we briefly present SPEM 2.0, the process modeling language used to model safety processes. In Section II-B we recall the basic principles of model-driven engineering. Finally, in Section II-C, we briefly present GSN and SACM, the graphical notation and its meta-model used to model safety arguments.

A. Process and SPEM 2.0-based process modeling

A process identifies a structure that is imposed on the development of a system. More precisely, a process can be defined as a set of partially ordered tasks that have to be executed to develop systems. The main process elements that can be associated to a task are: work-products (e.g., artifacts, deliverables, outcomes, etc.), roles, guidance (e.g., templates), tools, etc. Tasks can be grouped to form an activity and activities in turn can be grouped to form a phase. To model a process various languages are at disposal. In this paper, we select SPEM 2.0 (see [4], [5] for the motivation of this choice). SPEM (Software Process Engineering Meta-model) 2.0 [6] is the OMG's standard for systems and software process modelling. SPEM 2.0 offers static as well as dynamic modelling capabilities. In this paper, we mainly limit our attention to the static modeling. SPEM 2.0 offers support for the definition of reusable process content (MethodContent package). Process engineers are enabled to define reusable *work definition* elements (e.g. phases, activities, tasks, etc.) as well as elements representing: who is responsible for the work (roles), how the work should be performed (guidance), what should be expected as in/output (work-products) and which tool should be used to perform the work.

In Table I, we recall a subset of SPEM 2.0 modelling elements, which can be interrelated to model static process structures (except for *TaskUse* that can be used to model dynamic structures). More precisely, we only recall those elements that we use in Section III.

TABLE I. ICONS DENOTING METHOD CONTENT (USE) ELEMENTS

Task	TaskUse	Role	WorkProduct	Tool	Guidance
					

Despite their general-purpose nature, SPEM 2.0 modeling elements implicitly permit process engineers to model safety concerns. In [3], however, a SPEM 2.0 extension is proposed to model safety concerns (e.g. integrity levels) explicitly.

B. Model-driven Engineering

Model-driven Engineering (MDE) [7] is a model-centric software development methodology aimed at raising the level of abstraction in software specification and increasing automation in software development. MDE indeed exploits models to capture the software characteristics at different abstraction levels. These models are usually specified by using (semi) formal domain-specific languages. For automation purposes,

model transformations are used to refine models (model-to-model transformations) and finally generate code (model-to-code transformations). A model transformation (e.g. Model-to-Model) transforms a source model (compliant with one meta-model) into a target model compliant with the same or a different meta-model. Besides vertical transformations for software development, horizontal transformations can be conceived for other purposes (e.g. verification, etc.). A standard transformation can be defined as a set of rules to map source to the target. Each rule describes how to transform source instances to the identical target. Many languages are available to specify transformations. For instance, to specify Model-to-Model (M2M) transformations, declarative as well as operational languages can be used. Transformations are executed by transformation engines.

C. GSN

As already summarized in [8], to document safety cases, several approaches exist [9]. GSN [10] is one of them. GSN is a graphical notation, which permits users to structure their argumentation into flat or hierarchically nested graphs (constituted of a set of nodes and a set of edges), called goal structures. To make the paper self-contained, in Fig. 1, we recall the concrete syntax of the core GSN modelling elements used in Section IV. As Fig. 1 shows, all the nodes are characterized by an identifier (ID) and a statement, which is supposed to be written in natural language.



Fig. 1. Subset of GSN concrete syntax.

We recall that a *Goal* represents a claim about the system; a *Strategy* represents a method that is used to decompose a goal into sub goals; a *Solution* represents the evidence that a particular goal has been achieved; a *Context* represents the domain or scope in which a goal, evidence or strategy is given; *Supported by* represents an inferential (inference between goals) or evidential (link between a goal and the evidence used to substantiate it) relationship. Finally, *In context of* represents a contextual relationship.

ARgument Metamodel (ARM) [11] represented an effort to unify and standardize the graphical notations (namely GSN and CAE [12]) broadly used for documenting safety cases. By providing a meta-model that defines the abstract syntax of a unified argumentation language, ARM thus constitutes a step towards the formalization of these notations. The OMG specification provides tables that show the mapping between ARM concepts and GSN/CAE concepts. Columns 2-3 in Table II recall the mapping between ARM and GSN (focus on core elements). More recently, another OMG standard, called SACM, superseded ARM. SACM (Structured Assurance Case Metamodel) [13] combines ARM and Software Assurance Evidence Metamodel (SAEM) and preserves the mapping shown in Table II.

III. GENERATION AND REUSE OF PROCESS-BASED ARGUMENTS

In the context of safety certification, it is required to collect and structure the evidence that a system is acceptably safe. Generally, this requires the provision of process as well as product-based arguments. A safety case should be constituted of two branches (one devoted to process-based argumentation and the other to product-based argumentation). These branches could be developed in parallel and be inter-related. In some safety standards, these branches can be provided separately. As recalled in the introduction, within ISO 26262, the process-based argumentation is provided separately to be evaluated and documented within the Safety Functional Audit work-product. In this section, we focus on the process-based branch and we present a method to generate and reuse process-based arguments. In particular, in Section III-A we give an overview of our model-driven safety certification method. In Section III-A, we provide the conceptual mapping between SPEM 2.0 and ARM/SACM. Then, in Section III-C, we sketch in natural language the meaningful steps of the algorithm that should be executed to automatically generate process-based arguments from process models.

A. Model-driven Safety Certification

To generate certification artifacts, we propose to use MDE principles and apply them in the context of certification. The idea is to pioneer a Model-Driven Safety Certification (MDSafeCer) method enabling automatic generation of argumentation models from process models. The goal is not the creation of novel goal structures, but the generation of goal structure that have successful stories and a proven compelling power. Thus, reuse of experience is crucial to provide adequate transformation rules allowing for the generation of easy-to-maintain and easy-to-review arguments.

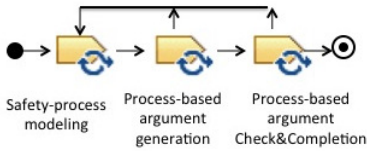


Fig. 2. MDSafeCer overview specified in SPEM 2.0.

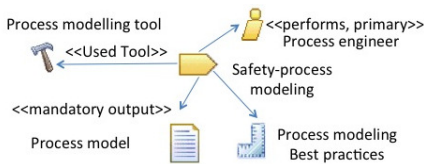


Fig. 3. Safety process modeling.

As Fig. 2 shows, MDSafeCer is constituted of three chained iterative tasks. The first task, called “Safety process modeling” is detailed in Fig. 3. This first task shows that a process engineer is responsible of modeling a safety process according to the best practices in process modeling as well as according to the standard(s). To model a process, a modeling tool is used.

As shown in Fig. 4, once the model is available the process engineer generates a process-based argument by using a model

transformation implemented within a transformation engine. As shown in Fig. 5, this argument, which can be considered a “raw” or better *defeasible* [14] argument, is then checked and eventually corrected (if fallacies are detected) and/or completed by a safety argumentation expert. Checking and completion is an iterative task, which takes in input also the feedback provided by external assessors. If the transformation engine or the safety argumentation expert detect problems related to the process-based argument due to e.g. missing/wrong information in the process model, new iterations of the first task are required.

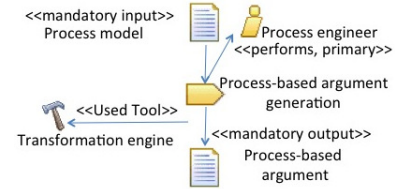


Fig. 4. Process-based argument generation.

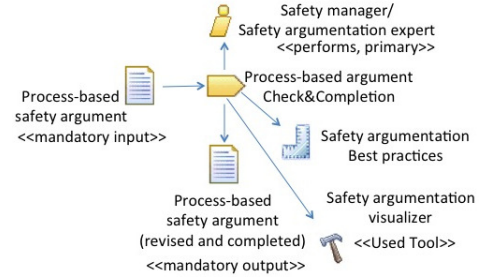


Fig. 5. Process-based argument Check&Completion.

To perform the generation of the process-based argument via model transformation, no constraint on the source and target meta-models exists. However, by considering the current state of the art in terms of standardization, tool-support and active research community, we choose SPEM 2.0 for the source space and ARM/SACM for the target space. Fig. 6 shows the M2M intended transformation. In case of more appropriate future alternatives, our general approach remains valid. As recalled in Section II, both SPEM 2.0 and ARM/SACM are two domain-specific meta-models and in the context of this paper they represent a possibility towards the realization of our MD-SafeCer method, allowing for the generation of ARM/SACM-compliant argumentation models from SPEM 2.0-compliant process models.

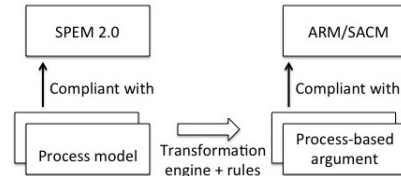


Fig. 6. M2M transformation.

As we discussed in [15] and as it was mentioned in [16], the goal of automation is not to replace human reasoning, but to focus it on areas where they are best used. Similarly, in this work we are not aiming at eliminating human reasoning

from the process of safety reasoning and argumentation, but to support it by providing automation of more clerical tasks. The human expertise on safety argumentation is still required to check and complete the argument obtained via model transformation and thus ensure that the final argument will be compelling, comprehensible and valid [17].

B. Mapping SPEM 2.0 and ARM/SACM Concepts

As recalled in Section II, crucial process elements can be associated to a process task ta , namely a set of roles R , a set of work-products W , a set of guidances G , and a set of tools T . In the context of process compliance certification, the execution of a process task constitutes a process-based sub-argument. Based on this statement, we can easily develop a mapping between a subset of SPEM 2.0 concepts and ARM/SACM concepts. From the presence of a task ta in a process model, the goal (sub-goal) “the task ta has been executed in compliance with the required standard” should be generated. The process elements associated to the task constitute the evidence (solution in GSN terms) within a process-based sub-argument that the task has been executed. Thus, from the presence of these associated process elements, appropriate solutions (information elements) should be generated and linked to the sub-goal by using supportedBy (an instance of the AssertedEvidence) links that relate solutions (information elements) with the sub-goal (claim). Table II summarizes this concepts mapping.

TABLE II. CONCEPTS MAPPING

SPEM2.0	GSN	ARM/SACM
Task ta	Goal	Claim
Role ro	Solution	InformationElement
Work product wp	Solution	InformationElement
Tool to	Solution	InformationElement
Guidance gu	Solution	InformationElement
Relationship between ta and $ro/to/wp/gu$	supportedby	AssertedEvidence

Moreover, if the process elements associated to ta are of different types (roles, work-products, etc.), then a strategy (ArgumentReasoning in ARM/SACM) should be included to argue over each single element within each set R , W , G , T . Finally, a relationship inContextOf (AssertedContext in ARM/SACM) should relate the sub-goal related to the task with a piece of contextual information related to the standard to be considered.

C. Process-based sub-arguments generation

The main goal of process-based argumentation is to show compliance with the mandated safety process(es). We propose a possible argumentation pattern constituted of a top level claim stating that “the process adopted is in compliance with the required standard(s)”. This claim can be decomposed by showing that all the process activities have been executed and that in turn for each activity all the tasks (belonging to the set $TASKS$) have been executed and so on until an atomic process-related work-definition unit is reached. This method supports compositional argumentation and reuse. In the case a task represents a commonality [4], its corresponding argumentation can be reused as it is and composed with other argumentation fragments.

In what follows, we sketch the rules conceived to generate in output the task-related sub-goal-structure. In input, the

algorithm takes: a process structure (more specifically, a task structure like the one shown in Fig. 7), the in-progress goal structure (ARM/SACM-compliant sub-graph), the connection points. The rules are given by following the same approach followed in [15]. We create a process-based argument-fragment for a process task ta by using the following rules:

1. Create the top-level goal ID:G1 and statement: “The task ta has been carried out”. Create the context to be associated to G1. Context ID:C1 and statement: “Standard $\{x\}$ ”, where x is a variable. Create an inContextOf link to relate G1 and C1.

Develop the goal G1 further by creating four strategies and for each strategy a set of sub-goals.

- (a) S1: “Argument over roles R ”.
- (b) S2: “Argument over work products W ”.
- (c) S3: “Argument over tools T ”.
- (d) S4: “Argument over guidance G ”.

2. Further develop strategy S1 and for every role ro in R : create a goal G1.ro “ ro is certified” and develop this goal further by creating the corresponding solution E.ro “ ro ’s certifications” and the supportedBy links necessary to link S1 with G1.ro and G1.ro with E.ro.

3. Further develop strategy S2 and for every work product wp in W : create a goal G1.wp “ wp is available” and develop this goal further by creating the corresponding solution E.wp “ $\{wp\}$ -related name” and the supportedBy links necessary to link S2 with G1.wp and G1.wp with E.wp.

4. Further develop strategy S3 and for every tool to in T : create a goal G1.to “ to is qualified” and develop this goal further by creating the corresponding solution E.to “ to ’s qualifications” and the supportedBy links necessary to link S3 with G1.to and G1.to with E.to.

5. Further develop strategy S4 and for every guidance gu in G : create a goal G1.gu “Guidance gu has been followed” and develop this goal further by creating the corresponding solution E.gu “ $\{gu\}$ where and how” and the supportedBy links necessary to link S4 with G1.gu and G1.gu with E.gu.

The rules are presented using GSN terminology to allow the reader to quickly see the generation of the GSN goal structure presented in Section IV. Other users might prefer using CAE instead of GSN. However, as discussed in the previous subsection, the mapping between GSN, CAE and ARM/SACM is clearly defined and our approach is based on ARM/SACM. As discussed in [9], non-graphically inclined modelers might prefer a textual concrete syntax instead of GSN. In such a case, ARM/SACM can be easily mapped to a textual representation of argumentation concepts, e.g., by paraphrasing GSN goal-structures. Beside personal preferences (image/word inclined) of MDSafeCer users, a documenting style may result more talkative when interacting with assessors. So, ideally, once the in-progress meta-models will be stable, a powerful MDSafeCer could support various transformations.

IV. APPLYING MDSAFECER

In this section, we apply MDSafeCer. First of all we act as process engineers and we model in SPEM 2.0 a process task

in compliance with ISO 26262. In particular, we consider the task Hazard identification, part of the Hazard analysis and risk assessment clause. Then, we apply the generation rules given in Section III and we obtain the corresponding process-based argumentation in GSN. The rules are performed manually but this task is expected to be tool-supported. Then, we play the role of the safety manager and we evaluate the quality of the generation.

A. Process Task Modelling

Fig. 7 shows how the task Hazards identification can be modeled in SPEM 2.0.

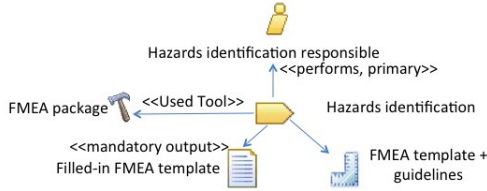


Fig. 7. Hazards identification in SPEM 2.0.

To the task “Hazard identification” (also names *ta1*) the following process elements are associated: a role (the responsible for hazards identification also named *ro1*), a work-product (Filled-in FMEA template also named *wp1*), a guidance (work-sheet and guidelines for Failure Mode and Effects Analysis, known as FMEA template, also named *gu1*), and a tool (FMEA software packages, also named *to1*).

B. Process-based Sub-argument Provision

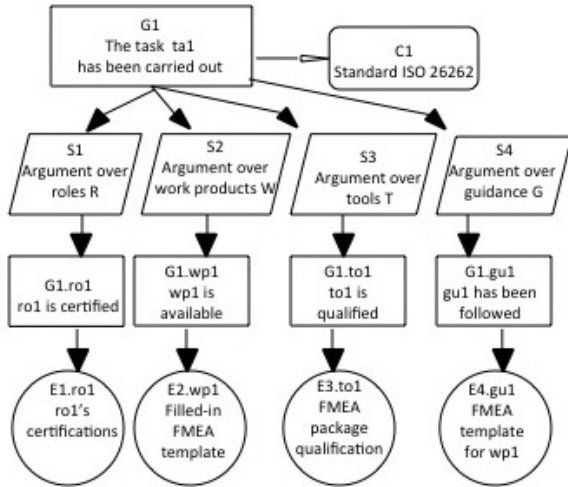


Fig. 8. Hazards identification in GSN.

On the basis of the information contained in the model (shown in Fig. 7) related to the task “Hazards identification”, by applying the rules given in Section III-C, the sub-goal-structure presented in Fig. 8 can be obtained. This task is required for all ASIL and also in other standards. As discussed in [4], prescriptive processes are mandated by various standards and the “Hazards identification” task can be considered a cross-domain commonality. The potential reusability of the process-based arguments is supported by a recent work [18],

which shows by modeling an automotive process line that in automotive standards common process elements are present. Thus, reuse in terms of process elements and process-based certification artefacts is a concrete possibility.

C. Process-based Sub-argument Evaluation and Completion

A safety argumentation expert at a first glance may be disappointed in front of this initial result. The fragment obtained via transformation rules, however, represents an interesting starting point. The rules permit the generation of fragments which contain the essential elements for process compliance. For sake of simplicity and space reasons, the rules are not particularly detailed and do not handle exceptions yet (e.g. missing information in the process model from which default GSN elements could be proposed). Thus, a safety argumentation expert could propose to introduce additional GSN modeling elements to: 1) denote parts that should be developed (e.g., by adding *Undeveloped goals* [10]) and 2) strengthen the confidence of the argument (e.g., by adding *Justification* [10], *Assumption* [10]).

Moreover, from a structuring point of view, the expert could suggest to improve the generation towards contract-based and modular structuring methods. Assuming for instance that tasks can be performed by different teams working in different departments, a contract-based approach would make sense since it would emphasize the contractual nature of the application of the development process. *Contracts* [10] could be added to join and organize the different process-based argumentation fragments generated by different teams. Furthermore, the evidence concerning the certification/qualification of the tool could be represented via a different modeling element (i.e., an *AwaySolution* [10]) to state that this evidence is provided elsewhere as a result of the tool qualification process (ISO 26262, Part 8.11). To ensure flexibility, various structures could be provided and offered as alternatives to satisfy different argumentation styles.

V. RELATED WORK

To ensure compliance as well as reduce time and cost, different solutions (compliance checking, reuse, automatic generation, etc.) are being investigated under different perspectives, mainly product-based perspectives. Exceptions to this product-based focus are the contributions presented in [19]. In [19], authors propose a workflow-based approach to provide: 1) reference models for the safety processes mandated by the standards and 2) automatic compliance checking capabilities of user-defined processes against reference models. As we discussed in [15], generating safety case arguments to increase efficiency of safety certification is becoming a hot research topic. A method for generating such arguments based on an automatic extraction of information from existing work-products is presented in [20]. The generated arguments consist of summaries of different work products created within a project. Similarly, a method for safety case assembly from process artefacts is presented in [21]. None of these methods, however, introduces a clear model-driven method. Meta-modeling in the context of safety cases is a rather recent research topic and thus the development of model-driven methods has been delayed due to the absence of standardized, stable and fully formalized meta-models. More recently, in [22], [23], we provided an

intuition on how families of safety cases could be generated from models related to process lines/families. This work differs from these ones since it focuses on sets of processes.

VI. CONCLUSION AND FUTURE WORK

To reduce cost and time during the certification process, in this paper, we have presented a novel model-driven method, called MDSafeCer, which permits users to generate process-based arguments from process models. In essence, MDSafeCer maps (ideally reusable) process structures onto (ideally reusable) argumentation structures (patterns). The method has been shown in the context of ISO 26262. However, what has been presented is valid also in the context of other standards. The method is also applicable for assurance cases and more generally trust cases, whenever process-based arguments play a significant role in the certification process. Concluding, our work offers a novel solution allowing for reusable, and (semi)-automatic derivable process-based arguments.

In a short-term future, in cooperation with industry, safety assessors and based on the state of the art (e.g., [24]), we plan to fully define a pattern for arguing about process compliance. Then, we plan to tune our rules to generate process-based arguments in compliance with our pattern. In a medium/long-term future, we plan to provide a prototype of tool-support. The idea is to exploit currently available open-source tools and provide a tool-chain. In particular, we intend to transform process models developed in EPF-Composer [25] into process-based sub-arguments to be given in input to GSN-compliant tools such as D-Case Editor [26]. The main goal of the prototype is to provide evidence with respect to the effectiveness of the approach in terms of time and cost reduction (manual vs. semi-automatic work). Once the evidence is achieved, the intention is to cooperate with industry to provide an industry-friendly tool support.

Acknowledgments: This work has been partially supported by the European Project ARTEMIS SafeCer [27] and by the Swedish SSF SYNOPSIS project [28]. We thank Henrik Thane (Safety Integrity AB) for fruitful discussions concerning the role of assessors in safety certification.

REFERENCES

- [1] ISO26262, "Road vehicles Functional safety. International Standard, November," 2011.
- [2] D. Jackson, M. Thomas, and L. I. Limmet, *Software for Dependable Systems: Sufficient Evidence?* Washington DC, USA: National Academy Press, 2007.
- [3] B. Gallina, K. R. Pitchai, and K. Lundqvist, "S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tunable Safety-oriented Processes," in *Proceedings of the 11th International Conference on Software Engineering Research, Management and Applications (SERA), Prague, Czech Republic, August 7-9, 2013*. Springer SCI, 2014.
- [4] B. Gallina, I. Slijivo, and O. Jaradat, "Towards a safety-oriented process line for enabling reuse in safety critical systems development and certification," in *Post-proceedings of the 35th Software Engineering Workshop (SEW-35), 12-13 October 2012*. IEEE, October 2012.
- [5] S. Kashiyarandi, "Reusing Process Elements in the Context of Safety Critical Systems Development and Certification," Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, Sweden, to appear.
- [6] Object Management Group, *Software & Systems Process Engineering Meta-Model (SPEM), v2.0. Full Specification formal/08-04-01*, 2008.
- [7] M. Biehl, "Literature study on model transformations." Embedded Control Systems. Royal Institute of Technology, Stockholm, Sweden, Tech. Rep. ISRN/KTH/MMK/R-10/07-SE, 2010.
- [8] 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)*, 2012, pp. 349–354.
- [9] C. Holloway, "Safety case notations: Alternatives for the non-graphically inclined?" in *Proceedings of the 3rd IET International Conference on System Safety*. IET Press, 2008, pp. 1–6.
- [10] GSN, "Community Standard Version 1," 2011.
- [11] ARM, "<http://www.omg.org/spec/arm/>."
- [12] L. Emmet and G. Cleland, "Graphical notations, narratives and persuasion: A pliant systems approach to hypertext tool design," in *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia*, ser. HYPERTEXT '02. New York, NY, USA: ACM, 2002, pp. 55–64.
- [13] SACM, "<http://www.omg.org/spec/sacm/1.0>."
- [14] J. Rushby, "Mechanized support for assurance case argumentation," in *Proceedings of the 1st International Workshop on Argument for Agreement and Assurance (AAA)*, ser. LNCS. Springer-Verlag, 2013.
- [15] I. Slijivo, B. Gallina, J. Carlson, and H. Hansson, "Generation of safety case argument-fragments from safety contracts," in *The 33rd International Conference on Computer Safety, Reliability and Security*, ser. LNCS, vol. 8666. Springer-Verlag, 2014, pp. 170–185.
- [16] J. Rushby, "Logic and epistemology in safety cases," in *Proceedings of the 32nd International Conference on Computer Safety, Reliability, and Security*, ser. LNCS, vol. 8153. Springer-Verlag, 2013, pp. 1–7.
- [17] I. Habli and T. Kelly, "Safety case depictions vs. safety cases - would the real safety case please stand up?" in *System Safety, 2007 2nd Institution of Engineering and Technology International Conference on*, 2007, pp. 245–248.
- [18] B. Gallina, S. Kashiyarandi, H. Martin, and R. Bramberger, "Modeling a safety- and automotive-oriented process line to enable reuse and flexible process derivation," in *8th IEEE International Workshop Quality-Oriented Reuse of Software*, July 2014.
- [19] P. W. H. Chung, L. Y. C. Cheung, and C. H. C. Machin, "Compliance flow - managing the compliance of dynamic and complex processes," *Know.-Based Syst.*, vol. 21, no. 4, pp. 332–354, May 2008.
- [20] E. Armengaud, "Automated safety case compilation for product-based argumentation," in *ERTS 2014: Embedded Real Time Software and Systems*, February 2014.
- [21] E. Denney and G. Pai, "A lightweight methodology for safety case assembly," in *Proceedings of the 31st International Conference on Computer Safety, Reliability and Security*, ser. LNCS, vol. 7612. Springer-Verlag, 2012, pp. 1–12.
- [22] B. Gallina, S. Kashiyarandi, K. Zugsbrati, and A. Geven, "Enabling cross-domain reuse of tool qualification certification artefacts," in *1st International Workshop on DEvelopment, Verification and Validation of cRiTical Systems, SAFECOMP Workshop*, ser. LNCS, vol. 8696. Springer-Verlag, 2014, pp. 255–266.
- [23] B. Gallina, K. Lundqvist, and K. Forsberg, "THRUST: A Method for Speeding Up the Creation of Process-related Deliverables," in *Proceedings of the 33rd IEEE Digital Avionics Systems Conference*, ser. DASC, 2014.
- [24] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, and R. Palin, "Safety cases and their role in iso 26262 functional safety assessment," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, F. Bitsch, J. Guiochet, and M. Kaniche, Eds. Springer Berlin Heidelberg, 2013, vol. 8153, pp. 154–165.
- [25] Eclipse Process Framework, "<http://www.eclipse.org/epf/>"
- [26] Y. Matsuno, "D-Case Editor: A Typed Assurance Case Editor," in *Proceedings of the 13th Real Time Linux Workshop, Prague, Czech Republic, October 20-22, 2011*.
- [27] ARTEMIS-JU-269265, "SafeCer-Safety Certification of Software-Intensive Systems with Reusable Components," 2013.
- [28] SYNOPSIS-SSF-RIT10-0070, "Safety Analysis for Predictable Software Intensive Systems. Swedish Foundation for Strategic Research."