

THRUST: A METHOD FOR SPEEDING UP THE CREATION OF PROCESS-RELATED DELIVERABLES

Barbara Gallina, Kristina Lundqvist, Mälardalen University, Västerås, Sweden

Kristina Forsberg, Saab AB, Jönköping, Sweden

Abstract

Certification of safety-critical avionics systems is an expensive and time-consuming activity due to the necessity of providing numerous deliverables. Some of these deliverables are process-related. To reduce time and cost related to the provision of process-related deliverables, in this paper, we propose to combine three approaches: the safety-oriented process line engineering approach, the process-based argumentation line approach, and the model driven certification-oriented approach. More specifically, we focus on safety-related processes for the development of avionics systems and we define how these three approaches are combined and which techniques, tools and guidelines should be used to implement the resulting approach, called THRUST. Advantages and disadvantages of possible existing techniques and tools are discussed and proposals as well as conceptual solutions for new techniques are sketched. Based on the sketched conceptual solutions, we then apply THRUST to speed up the creation of process-related deliverables in compliance with DO-178B/C.

Introduction

Certification of safety-critical avionics systems is an expensive and time-consuming activity due to the necessity of providing numerous deliverables. Some of these deliverables are process-related and they are aimed at either planning the activities to be carried out to meet the objectives or showing compliance with the guidelines and/or recommended practices. To develop the software for avionics systems, for instance, based on DO-178B/C, a generic process might be sketched. A company might provide its interpretation of the de-facto standard and create a process model under the assumption that “one size fits all”. Experience, however, teaches that a generic process is not meaningful enough and that processes tend to vary due to several factors (e.g. version of the standard, business area, development stage, software level, project-specific requirements,

geographical location, etc.). As a consequence of these variable factors, some parts of the process-related deliverables (e.g. plans for stating the intentions and arguments for showing compliance) have to vary consistently. Since, due to these variable factors, one size does not fit all (i.e. various processes and corresponding plans and compliance-related assurance cases have to be provided), the focus moves from single processes and related information to sets of processes that may exhibit reusable similarities and differences. Currently, no satisfying method exists to exploit similarities and thus enable reuse of process-related information. Another relevant consideration is that process-related deliverables exhibit a rather standardized structure and include information that could easily be semi-automatically generated from process models. Currently, however, no support exists to perform this.

To reduce time and cost and at the same time to increase quality related to the provision of process-related deliverables, the adoption of the safety-oriented process line approach [1, 2] permits process engineers to identify, based on their experience, common and variable certifiable process elements to be selected and composed to achieve flexible processes. A process activity for instance can either reflect the best interpretation of the facto standard or represent a valuable and equivalent alternative according to the standard tailoring rules. By identifying and modelling such process information, systematic reuse is enabled and thus time and cost can be reduced significantly. Similarly, the adoption of the “process-based argumentation line” [3, 4] permits safety managers to identify, based on their experience, common and variable process-based arguments to be selected and composed to achieve flexible arguments for process compliance. To achieve the same objectives (time and cost reduction & quality increase), the adoption of the model driven engineering (MDE)-oriented approach is also effective. The effectiveness is motivated by the fact that an MDE-oriented approach permits process engineers to avoid wasting time while providing

process-related information. Adequate tool-supported model transformations enable the (semi) automatic generation of such information. Process engineers and safety managers still play a crucial role but their time can be dedicated to the manual production of portions of deliverables that strictly require human intervention. In [5] as contribution to [6], a method called MDSafeCer is proposed and pioneers the adoption of MDE for certification purposes. This method, however, does not consider sets (families/"lines").

Thus, to reduce time and cost related to the provision of process-related deliverables, in this paper, we propose to combine the safety-oriented process line engineering approach, the process-based argumentation line, and the model driven certification-oriented approach. More specifically, we focus on (lines of) safety-related processes for the development of avionics systems and we define how these three approaches are combined and which techniques, tools and guidelines should be used to implement the resulting approach, called THRUST. Advantages and disadvantage of possible existing techniques and tools are discussed and proposals as well as conceptual solutions for new techniques are sketched. Based on the sketched conceptual solutions, we apply THRUST to speed up the creation of DO-178B/C-compliant process-related deliverables. Based on the results obtained by applying THRUST, we provide our lessons learned.

The rest of the paper is organized as follows. In the background section, we provide some fundamental information onto which the presented work is based. The core section presents THRUST; followed by a section aimed at showing the usage and potential effectiveness of THRUST. This is done by applying THRUST for the creation of a portion of DO-178B/C-compliant deliverables. Based on the application of THRUST, we derive our lessons learned. Finally, some concluding remarks and suggestions for future work are presented.

Background

This section recalls the essential background on which the presented work is based: DO-178B/C, safety-oriented process lines engineering, process modeling, safety-oriented process line modeling, process compliance, process compliance

documentation, model-driven engineering/certification.

DO-178B/C

DO-178C [7] has been replacing its predecessor (DO-178B [8]), which has represented the de facto standard in the avionics domain for a couple of decades. DO-178C addresses the inconsistencies of the previous document but preserves its basic and valuable principles. DO-178C, as its predecessor, provides guidance for the development of software for airborne systems and equipment. Its purpose is to guarantee a level of confidence in the correct functioning of the software developed in compliance with airworthiness requirements. To do that, it provides a series of processes characterized by a set of objectives, activities and expected deliverables. Process planning is one of these processes. Among its expected deliverables we have: software development plan (SDP) and Plan for Software Aspects of Certification (PSAC). In the context of this paper we will focus on these two deliverables. SDP is a plan, which provides a detailed description concerning how the software should be developed. More specifically, SDP includes: a) the identification of standards, b) the software life-cycles (requirements process, design process, etc.), and c) the software development environment. Taken altogether, a-c detail the software process. SDP can be included within PSAC, which, as stated in [7], serves as the primary means for communicating the proposed development methods to the certification authority for agreement and defines the means of compliance with DO-178B/C. PSAC includes the software life-cycle, the software life-cycle data, plus various other items that are not in focus in this paper.

Within an SDP, a design process could be characterized by:

Input: Software development plan, Software Requirements Data, Software Design Standards.

Output: Design description.

Roles: designers in charge of the design decision related to functional requirements and quality (safety) experts in charge of the design decision related to non-functional requirements.

Guidelines: guidelines, defined in Section 5.2.2 of the standard, contain general as well as safety specific information.

Tools (company-specific decision): Unified Modeling Language (UML) and a model-based development environment (e.g., SCADE Suite).

This design process may vary due to the software level, whose variation constrains other variabilities, as specified in Annex A in [7].

Safety-oriented Process Lines Engineering

Safety-oriented process lines [1] represent sets of safety-oriented processes that may exhibit: full commonalities (equal process elements), partial commonalities (structured process elements that are partially equal), and variabilities. Variabilities denote elements that may vary e.g., optional process elements or process elements that represent variants and can be chosen instead of others at specific variation points. The fundamental process elements to be interconnected to model processes are: tasks (which represent broken down units of work), work products (e.g., deliverables), roles, guidance, and tools.

As recalled in [1], safety-oriented process lines can be engineered by adopting a three-phase approach consisting of a first phase aimed at scoping the process line, a second phase aimed at engineering the domain (i.e., modeling (partially) common and variable process elements) and a third phase aimed at engineering the single processes by selecting and composing reusable process elements to obtain the models related to single processes. To show the potential for intra as well as cross-domain reuse, in [2], we have engineered an automotive safety-oriented process line constituted of development processes while in [3] we have engineered a cross-domain safety-oriented process line constituted of tool qualification processes.

Safety-oriented Process Line Modeling

As we discussed in [1, 2], to model processes, various general-purpose languages are at disposal e.g., SPEM 2.0 [9]. However, currently, no language is available to model safety-oriented process lines. In [2], due to the necessity of having a tool at disposal, we proposed a methodological approach to model safety-oriented process lines in EPF-Composer [10] via some workaround solution.





Recently, two relevant extensions of SPEM 2.0 have been proposed: vSPEM [11], to model process lines and S-TunExSPEM [12] to model and exchange

safety-oriented processes (focus on DO-178B/C processes). However, no tool support exists for modeling by using these extensions. In our context, a combination of these two extensions could represent an interesting solution. S-TunExSPEM, for instance, could be extended with vSPEM constructs. Thus, in this subsection, we recall essential information related to these extensions. More specifically, with respect to S-TunExSPEM, we partially recall its safety-tunability, which is supported by the following language constructs:

- Safety-related process elements e.g., SafetyRole, SafetyTask, etc. These elements are characterized by the presence of a safety hats.
- An attribute to allow process engineers to set the safety level. Only four levels are at disposal since in case of negligible (e.g. no effect, level E in DO-178B/C) consequences related to the hazards, no specific safety-related process elements are needed. This attribute is syntactically concretized via the colour of the safety hat (i.e. red for the most critical safety level, followed by orange, yellow and bitter lemon). A red hat that decorates a role denotes high qualification (i.e., high level of proven experience and sufficient seniority to be considered competent and accountable for the actions the role is responsible for).



The above language constructs are concretized via the icons given in Table 1. Table 1 mainly (except for phase) shows the icons that can be used to define statically the processes. To define processes dynamically, additional icons are available. These additional icons are obtained by decorating SPEM2.0 *inUse* icons, in a similar way as for *Definition* icons.

Table 1. Subset of S-TunExSPEM Icons

Task	Role	Tool	Work product	Guidance	Phase
					

With respect to vSPEM, we recall its support for variability by focusing on the concrete syntax. As Table 2 shows, vSPEM basically introduces the possibility to model: 1) variation points, by decorating SPEM2.0 icons with empty circles; and 2) variants, by decorating SPEM2.0 icons with a V.

Table 2. Subset of vSPEM Icons

Concept	Variation point	Variant
Task		

To connect a variant (optional/alternative/etc. process element) to a variation point, vSPEM provides the *occupation relationship arrow*, which is an arrow having a filled circle on the opposite side.

Process Compliance

Safety certification requires the applicant to show that the product (e.g., aircraft) behaves acceptably safe and that the development process meets the objectives. Despite the absence of scientific evidence concerning the real efficacy of the development processes defined within the standards [13, 14 and 15], compliance with those processes is required. To be compliant, in general, a company has two alternatives. The first alternative consists of the strict and almost literal implementation of the process. This entails:

- the identification and assignment of roles/responsibilities.
- the execution of all the activities according a specific order (if any) and/or grouping (if any);
- the consumption/provision of all the required work products;
- the application of specific guidance (if any);
- the usage of specific tools (if any).

Each of the above steps has to be performed with the stringency required by the software level.

The second alternative consists of the execution of a tailored process obtained by applying tailoring rules (e.g. the usage of alternative methods/guidance if accepted by assessors) to the prescriptive one.

In the context of objective-based standards (e.g., DO-178B/C), processes are not prescriptive. The manufacturer has only to show that the objectives have been met.

Process compliance documentation

To document process compliance, two strategies may be adopted: single-process-centered, process-line centered. To avoid re-inventing the wheel,

whenever argumentation lines can be identified, the process-line centered alternative is preferable.

To document/argue about process compliance, various means are at disposal [16]: textual languages (e.g. semi-structured natural language), graphical languages (e.g., Goal Structuring Notation (GSN) [17], Claim-Argument-Evidence (CAE) [18]), or a combination of both. These means are more generally used to document safety cases. Recently, an OMG standard, called SACM [19], has been provided to unify argumentation languages (namely, GSN and CAE). GSN is currently the only documentation means that offers constructs to argue about argumentation lines [4]. Thus, in this subsection, we briefly recall its concrete syntax.

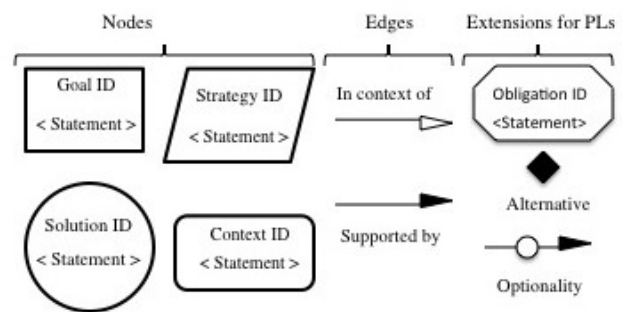


Figure 1. GSN Modeling Elements

The GSN modeling elements can be composed to structure the argumentation into flat or hierarchically nested graphs (constituted of a set of nodes and a set of edges), called goal structures. Of particular interest in the context of this paper is the possibility to document extrinsic variability i.e., the variability within the goal structure due to the variability within the process line model. A choice during the configuration of a single process, will have an impact on the goal structure. The interested reader may refer to [4 and 17] for a complete introduction of GSN and its extension.

Model-driven Engineering/Certification

As we recalled in [5], Model-driven Engineering (MDE) 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. 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. Besides vertical transformations for software development, horizontal transformations can be conceived for other purposes (semi-automatic generation of certification artefacts in the context of this paper).

THRUST

In this section we present our proposal, called THRUST. THRUST is a method that allows users to speed up the creation of process-related deliverables by combining safety-oriented process line engineering, process-based argumentation line engineering and model driven certification. Figure 2 provides an overview of THRUST given by using S-TunExSPEM. As Figure 2 shows, THRUST consists of four phases: two process-centered phases and two process-based argumentation-centered phases. The red hat is meant to highlight that THRUST is highly critical. THRUST is expected to provide support for efficient creation and management of process-related deliverables compliant with the most stringent development assurance level, A, of DO-178B/C.

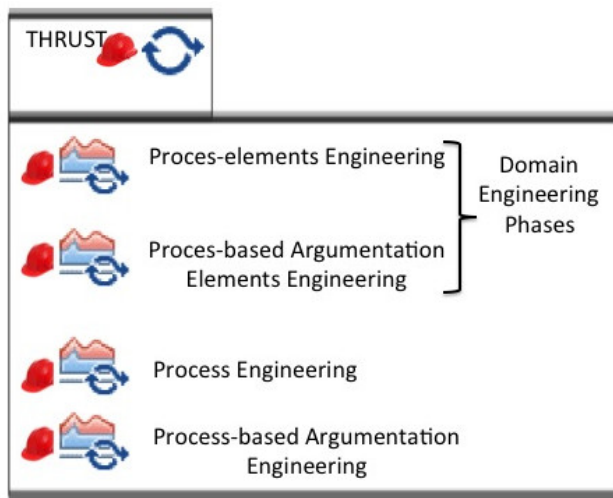


Figure 2. THRUST Overview

As Figure 3 shows, these phases can be partially ordered logically. Various executions are possible since concurrent phases can be serialized in various ways. At the end, however, the deliverables of both branches (left and right) have to be available to satisfy the certification authorities.

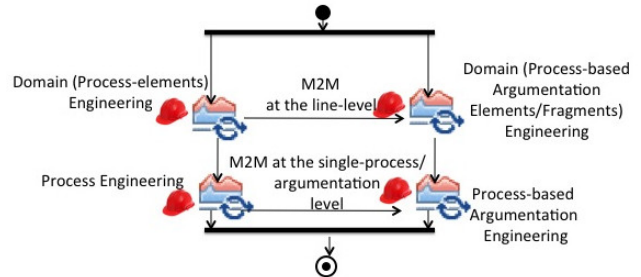


Figure 3. THRUST-high-level Activity Diagram

Process-centered phases

In this subsection, we focus on the left-hand branch of the activity diagram, shown in Figure 3, and we reveal the tasks that are embraced by the phases. As Figure 4 shows, the Domain (Process-elements) Engineering phase consists of three tasks, which can be iterated if needed. The first task consists of the interpretation of the set of standards according to S-TunExSPEM i.e., identification of tasks, safety tasks, etc. Then, common and variable process elements are identified. Finally the process line is modeled by using S-TunExSPEM extension.

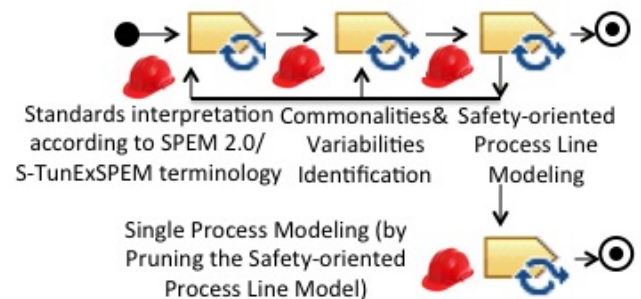


Figure 4. Focus on the Process-related Phases

Once the safety-oriented process line model is at disposal, single processes can be derived from it by selecting and composing the required process elements. This derivation is performed during the Process Engineering Phase.

Process-based argumentation-centered phases

In this subsection, we focus on the right-hand part of the activity diagram, shown in Figure 3 and we reveal the tasks that are embraced by the phases. As Figure 5 shows, similarly to the Domain (Process-elements) Engineering phase, the Domain (Process-based Argumentation elements/Fragments) Engineering phase consists of three tasks, which can be iterated if needed.

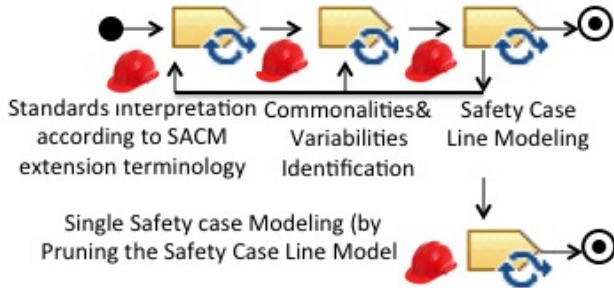


Figure 5. Focus on the Process-based Argumentation Phases

The first task consists of the interpretation of the set of standards according to SACM i.e., identification of goals (not a straightforward task as argued in [13]), evidence, etc. Then, common and variable process-based argumentation fragments are identified. Finally the process-related portion of the safety case line is modeled by using SACM extension.

Once the safety case line model is at disposal, single process-based arguments can be derived from it by selecting and composing the required process-based argumentation elements. This derivation is performed during the Process-based Argumentation Engineering Phase.

Model-driven Process-related Certification

The right-hand and the left-hand parts of the activity diagram can be combined via the model driven engineering approach. Model transformations can be performed either at the level of the individuals or at the level of the family/line. In [5], we have explored how model transformations could be performed at the level of individuals (single processes/single safety cases). In [3], we have given an intuition on how pattern-based safety case lines could be derived from safety-oriented process lines. In this paper, we are interested in pioneering the

research direction related to how model transformations could be performed at the line-level.

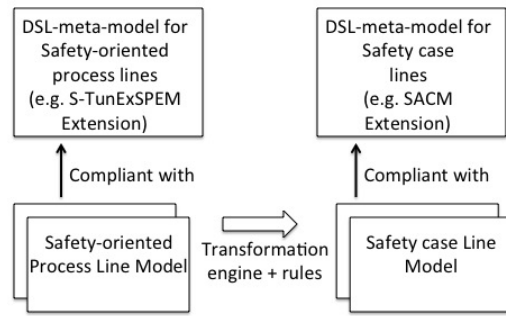


Figure 6. M2M-Model-to-Model Transformation

As Figure 6 shows, proper meta-models are needed. More specifically, two meta-models are needed: one Domain Specific Language (DSL) meta-model for safety-oriented process lines (source space) and another DSL meta-model for safety case lines (target space). Once these meta-models are at disposal, via model transformations, it is possible to generate models in compliance with e.g., a SACM extension from a model in compliance with e.g., S-TunExSPEM extension.

Despite the current absence of these meta-models, based on the background information (the not-yet formalized notations), we provide an intuition on how this could happen.

Process-based argumentation at the line level is currently not applied. However, we can reasonably state that its goal is to show compliance with the family of safety processes, mandated by the standards. Reasonably, the top-level claim could state that the modeled process line is in compliance with the required standard(s). This claim could be decomposed into sub-goals to show that all the process activities (including its partial commonalities and variants) have been modeled. Then, in turn, for each activity new subgoals could be introduced to show that all the tasks have been modeled and so on until an atomic process-related work-definition unit is reached.

In what follows, we sketch the rules for generating in output the task line-related sub-goal-structure. For sake of simplicity, our task line is rather limited. Its tasks may only vary based on the software level and based on company-specific optional tools. All the other process elements are considered as commonalities. As mentioned in the

introduction, various factors may play a role and may make a task vary. The purpose of this section, however, is not to show the entire spectrum of possible variations but simply to show what can be done to speed up the creation of process-related certification artefacts, if commonalities and variations are properly modeled.

In input, the algorithm takes: a process structure (more specifically, a task line structure like the one shown in Figure 7), the in-progress goal structure, the connection points. The rules are given by following the same approach followed in [5 and 20]. We create a process-based argument-fragment for a process task-line *tl* by using the following rules:

1. Create the top-level goal ID: *G1* and statement: “The task line *tl* has been planned with adequate stringency in accordance with the software levels”. For sake of clarity, it should be stressed that software levels are derived from DALs, which define the assurance levels as product (system) line level. Once *G1* is created, create the context to be associated to *G1*. Context ID: *C1* and statement: “Standard(s) {*x*}”, where *x* is a variable denoting a set of standards (a singleton is also a valid value for *x*). Create an *inContextOf* link to relate *G1* and *C1*.

Develop the goal *G1* further by creating:

- (a) either an alternative-related diamond plus an obligation element (connected to the diamond) stating the condition for the branching, in the case alternative tasks are present.

- (b) or an optional related arrow plus an obligation element (connected to the arrow) stating the condition for the option, in the case optional tasks are present.

2. If the diamond exists, further develop it and for every alternative task: create a goal *G1.ta* “*ta* has been planned with adequate stringency”. Connect this goal to the diamond. Then, develop this goal further. At this point, other rules similarly to what presented in [5, 6] should be added to consider common and variable process elements that can be connected to a task: roles, tools, work products, guidance.

3. If the optional related arrow exists, further develop it and for the optional task: create a goal *G1.top* “*top* has been planned with adequate stringency” and develop this goal further similarly to what presented in [5, 6].

Crosscutting constraints should also be defined.

Applying THRUST: an intuition

In this section we give an intuition concerning the application of THRUST at the line level. As mentioned previously, to apply THRUST various executions are possible. In [5], we have explored the execution of the bottom part of the activity diagram (Figure 3). More specifically, from single process models, we have manually obtained argumentation fragments related to single processes. In this paper, we focus on the upper part of the activity diagram and we focus on “lines” to show how a family of software development plans can be transformed into a family of arguments that show that the objectives stated in DO-178B/C are met. More specifically, we model a portion (focus on design) of the safety-oriented process line constituted of the processes described in corresponding SDPs. From this model we manually derive a GSN fragment that can be used to show that the safety process line is in compliance with DO-178B/C.

Modeling the design-related task line

In this subsection, we model by using a combination of S-TunExSPEM and vSPEM a design-related and safety-oriented task line (which was described in the background in natural language), part of a family of SDPs. For sake of simplicity, this task line is characterized by only 4 tasks. Figure 7 shows our task line. The root element is a task since the family/line modeled in Figure 7 is constituted of tasks.

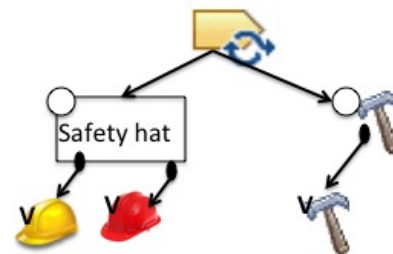


Figure 7. Design-related Task Line

As Figure 7 shows, our task line may vary due to the software level (which we assume can take only 2 values in the context of this paper due to the assumed product line). Note that the rectangle with the empty circle is a language construct temporarily proposed to denote a safety hat-related variation point. Our task

line also may vary at the company specific-level tool (SCADE is optional). Even if not shown in Figure 7, to the task, other process elements are associated: two roles (designer ro1 and safety manager ro2), three work-product in input (named wpi1, wpi2, wpi3), two guidance (named gu1 and gu2), and two tools (UML and SCADE, also named to1 and to2), one of which is optional.

Generating the design-related process-based argumentation line

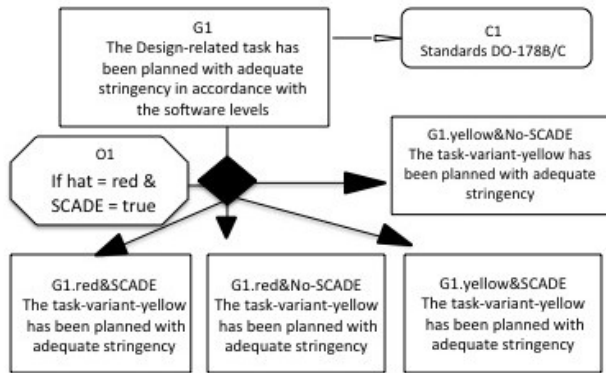


Figure 8. Goal Structure Fragment

On the basis of the information contained in the model (shown in Figure 7) related to the design-related task line, by applying the transformation rules, the sub-goal-structure presented in Figure 8 can be obtained, which shows a fragment of a process-based argumentation. This generated fragment does not pretend to be complete. Meaningfulness is also not one of its characteristics. The main intention is to show how the machinery could work.

Lessons learned

From the application of THRUST, despite the simplicity of our illustration, we can draw the following lessons.

General soundness- The THRUST approach is sound since commonalities and manageable variabilities can be identified and modeled. THRUST is also beneficial since commonalities enable reuse. The semi-automatic generation is desirable. Anyway a lot of work is required to properly ponderate and achieve the right trade off between what can be automatized and trusted and what should remain manual work. Due to the current limitations in terms of modelling and meta-modelling a lot of work is also required to achieve proper modeling and meta-

modeling means and corresponding tool-support. Thus, currently, THRUST has a great potential but its applicability is far from being satisfactory and scalable in industrial settings.

Related Work

As far as we know, currently there are no other approaches that aim at reusing both process elements and process-based arguments within a “line” perspective. Thus, due to the novelty of our approach, we split the discussion of the related work by considering separately works that have tried to achieve reusable processes and work that have tried to semi-automatically generate information needed for certification purposes. To reuse process elements, process lines are not the only possibility. In [21], a model-driven-based tailoring method has been proposed. Authors consider a generic process as a starting point and then propose to make it vary according to the needs via model transformations. With respect to our purposes and based on the current industrial needs (especially coming from product lines manufacturers), we believe that our method based on safety-oriented process lines is more suitable since the solution space (family-oriented) is closer to the problem space (also family-oriented). In [22], authors provide detailed guidelines useful to deal with legacy avionic software, which was certified by using a superseded version. Authors perform a comparative study between DO-178A and DO-178B and textually in natural language they describe what varies. Our approach builds on top of comparative studies (possibly performed by experts). Its ambition is however to enable semi-automatic generation and reuse thanks to (semi) formal machine-readable models. The relevance of organizing processes for reuse constitutes also a research theme in [23]. Some research questions proposed in [23] are also part of our research agenda.

As we already discussed in [5 and 20], currently, semi-automatic generation of certification artefacts is limited due to the limitations of the current status in terms of meta-models and formalized notations. Dialects of GSN are implemented and based on these dialects, mainly product-based GSN fragments are derived from verification analysis results.

Conclusion and Future Work

To ensure the safety of safety-critical systems, compulsory as well as advisory safety standards have been issued. Some of these standards define (prescriptive/objective-based) safety-oriented processes. Compliance with the standards is necessary to provide process-based evidence for certification purposes. To support efficient as well as cost-effective provision of safety-related process artefacts, we have introduced a new method, called THRUST. THRUST combines model-driven certification, safety-oriented process line engineering and process-based argumentation line engineering. Via this combination, THRUST enables reuse and semi-automatic generation and thus supports efficient and cost-effective provision of artefacts. For illustration purposes concerning the usage and effective potential, we have then applied THRUST for the provision of a small portion of DO-178B/C deliverables. From this application, we have drawn our lessons concerning the general soundness of THRUST.

In the future, we aim at further developing THRUST. More specifically, in the short-term future, we intend to experimentally validate our approach. Based on the work done in the framework of SafeCer [6 and 25], the idea is to define a set of research questions (e.g. is our method (focus on MDSafeCer) less time-consuming? Is reuse really possible? Does our method increase quality e.g., by reducing the potential fallacies in compliance-related assurance case?) to be investigated based on a more complex case-study. The goal of the validation is to provide evidence that THRUST/MDSafeCer is sound, feasible and applicable. Considering the complexity of the certification process, the concretization of THRUST will bring advantages in industrial settings. Then, in the medium-term future we aim at contributing to the provision of adequate meta-models to enable model driven certification at lines level. Our plan in this direction is to target the provision of a meta-model for safety-oriented process lines. Our intention is to build on top of S-TunExSPEM and extend it by integrating the variability support proposed in vSPEM.

References

- [1] Gallina, B., and I. Sljivo, and O. Jaradat, 2012, Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification. Post-proceedings of the 35th IEEE Software Engineering Workshop (SEW-35), IEEE Computer Society, ISBN 978-1-4673-5574-2, Heraclion, Crete (Greece).
- [2] Gallina, B., and S. Kashiyaandi, and H. Martin and R. Bramberger, 2014, Modeling a Safety- and Automotive-oriented Process Line to Enable Reuse and Flexible Process Derivation. Proceedings of the 8th IEEE International Workshop on Quality-Oriented Reuse of Software (QUORS), IEEE Computer Society, Västerås (Sweden).
- [3] Gallina, B., and S. Kashiyaandi, and K. Zugsbrati and A. Geven, September 8, 2014, Enabling Cross-domain Reuse of Tool Qualification Certification Artefacts. Proceedings of the 1st International Workshop on DEvelopment, VERification and VALidation of cRITical Systems (DEVVARTS), Springer, LNCS, Florence (Italy).
- [4] Habli, I., and T. Kelly, 2010, A Safety Case Approach to Assuring Configurable Architectures of Safety-Critical Product Lines. Proc. of the International Symposium on Architecting Critical Systems (ISARCS), Prague, Czech Republic, Springer, pp. 142-160.
- [5] Gallina, B., 2014, A Model-driven Safety Certification Method for Process Compliance. 2nd IEEE Workshop on Assurance Cases for Software-intensive Systems (ASSURE), Naples, Italy, 3-6 November, 2014. (under evaluation)
- [6] Gallina B. et al, 2014, nSafeCer, D121.1: Generic process model for integrated development and certification.
- [7] RTCA DO-178C (EUROCAE ED-12C), November 2011, Software Considerations in Airborne Systems and Equipment Certification.
- [8] RTCA DO-178B (EUROCAE ED-12B), 1992, Software Considerations in Airborne Systems and Equipment Certification, Washington DC.
- [9] OMG, 2008, Software & systems Process Engineering Meta-model (SPEM), v 2.0. Full Specification formal/08-04-01, Object Management Group.

- [10] Eclipse Process Framework www.eclipse.org/epf/
- [11] Martinez-Ruiz, T., and F. Garcia, and M. Piattini, 2014, Towards A SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. Book Chapter, DOI: 10.1007/978-3-540-70561-1_9.
- [12] Gallina, B., and K. R. Pitchai and K. Lundqvist, 2014, S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tuneable Safety-oriented Processes. 11th International Conference on Software Engineering Research, Management and Applications (SERA), SCI 496, Springer, ISBN 978-3-319-00947-6, Prague, Czech Republic, August 7-9, 2013.
- [13] Rushby, J., 2011, New Challenges In Certification For Aircraft Software. Substantially revised version of the paper included in the Proceedings of the Ninth ACM International Conference On Embedded Software (EMSOFT), Taipei, Taiwan, October, pp. 211-218.
- [14] Daniels, D., May, 13, 2014, The Efficacy of DO-178B. Proceedings of the first workshop on Planning the Unplanned Experiment: Assessing the Efficacy of Standards for Safety Critical Software (AESSCS).
- [15] Fusani, M. and G. Lami, 2014, On the efficacy of safety-related software standards. Proceedings of the first workshop on Planning the Unplanned Experiment: Assessing the Efficacy of Standards for Safety Critical Software (AESSCS), Newcastle upon Tyne, 13 May 2014.
- [16] Holloway, C., 2008, Safety case notations: Alternatives for the non-graphically inclined? In Proceedings of the 3rd IET International Conference on System Safety, IET Press, pp 1–6.
- [17] GSN. Community Standard Version 1. November, 2011, [http://www.goalstructuringnotation.info/documents/GSN Standard.pdf](http://www.goalstructuringnotation.info/documents/GSN%20Standard.pdf).
- [18] Emmet, L., and G. Cleland, 2002, Graphical notations, narratives and persuasion: A pliant systems approach to hypertext tool design. In Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, HYPERTEXT, New York, NY, USA, ACM, pp 55–64.
- [19] SACM <http://www.omg.org/spec/sacm/1.0>.
- [20] Sljivo, I., and B. Gallina, and J. Carlson, and H. Hansson, September 2014, Generation of safety case argument-fragments from safety contracts. In The 33rd International Conference on Computer Safety, Reliability and Security (SafeComp).
- [21] Hurtado Alegría, J. A., and M. C. Bastarrica, A. Quispe, S.F. Ochoa, 2014, MDE-based process tailoring strategy. Journal of Software: Evolution and Process, VL-26, IS-4, SN-2047-7481, pp. 386-403.
- [22] Marquez, J. C., 2011, Modification to Legacy Software Developed per DO-178A Level 1 to DO-178B Level A: How to Organize Software Life Cycle Data for Software Approval in Aircraft Certification. In: Latin American Symposium On Dependable Computing (LADC), São José dos Campos.
- [23] Rombach, D., and R. Jeffrey, B. Peterson, M. D’Ambrosia, M. Fusani, H.-W. Jung, S. Ferber, J.Münch, and A. Ocampo, 2006, Process Engineering. In “A Process Research Framework”, Eileen Forrester ed., Software Engineering Institute, pp. 20-28.
- [24] SYNOPSIS-SSF-RIT10-0070: Safety Analysis for Predictable Software Intensive Systems. Swedish Foundation for Strategic Research.
- [25] ARTEMIS-JU- 295373 nSafeCer - nSafety Certification of Software-Intensive Systems with Reusable Components.

Acknowledgements

This work has been partially supported by the Swedish SSF SYNOPSIS project [24] and by the European ARTEMIS nSafeCer project [25].

*33rd Digital Avionics Systems Conference
October 5-9, 2014*