A Model-Driven Engineering Framework to Support the Functional Safety Process

Bart Meyers*, Klaas Gadeyne*, Bentley James Oakes†,
Matthias Bernaerts†, Hans Vangheluwe†, and Joachim Denil†
*CodesignS, Flanders Make, Belgium
†AnSyMo/CoSys, University of Antwerp and Flanders Make, Belgium
{Bart.Meyers, Klaas.Gadeyne}@flandersmake.be,
{Bentley.Oakes, Hans.Vangheluwe, Joachim.Denil}@uantwerpen.be, Matthias.Bernaerts@student.uantwerpen.be

***Abstract*—The design of safety-related systems traditionally has long and costly development cycles due to the highly manual safety engineering process, which is guided by industry standards. In this paper, we present a modelling framework that supports DevOps principles of continuous testing and fast development iterations for the design of safety-critical systems. We show how modelling can help introducing DevOps in the context of functional safety analysis, and we also report how DevOps was used during the development of the framework.**

I. Introduction

Today, the automotive industry develops products which contain tens of millions of lines of code [1], and which combine mechanics and (software-driven) electronics. These enormously complex systems must place paramount importance on their functional safety to prevent injury or death. In general, this functional safety is achieved by reducing or removing the risk of hazards caused by system malfunctions. Manufacturers apply functional safety standards to ensure that safety integrity levels are met. The ISO 26262 standard [2] defines functional safety for all Electric/Electronic/Programmable Electronic (E/E/PE) safety-related systems in road vehicles, throughout their lifecycle. Today, automotive manufacturers typically require their suppliers to apply the standard.

The DevOps methodology aims for short design/testing cycles by providing tool support for continuous integration, continuous testing, early analysis, etc. In this context, current practices for functional safety analysis in the automotive sector contradict the DevOps methodology, due to the functional safety engineering process that needs to be followed in parallel with the development process. During the functional safety process, a safety case is constructed for systems whose malfunction has the potential to lead to an unreasonable level of risk. Safety requirements must be documented in a precise manner to be complete and satisfied, as in this 147-page report on functional safety requirements for a hydraulic braking system [3]. During development of the system, evidence and arguments are collected, e.g., by conducting hazard analyses and risk assessments (HARA) in the concept phase. As the process of functional safety analysis is primarily manual, involving documents, spreadsheets, meetings, etc. these artefacts need to be kept consistent, resulting in a long and costly functional safety process. Early adopters move to semi-formal notations such as SysML [4], but inconsistencies still occur. While the development process is highly iterative, the functional safety engineering process has become the major bottleneck for applying redesigns or changes to the concept (i.e., iterations in the development process), hindering the benefits of following the DevOps methodology.

This paper introduces a framework for the ISO 26262 standard that supports continuous testing and analysis, allowing users to keep artefacts consistent and perform automated analysis. This consequently shortens the design and testing cycle for safety engineers, and allows for a greater understanding of the system's behaviour once implemented. Our framework is based on a formal modelling language and a contract-based requirements language, both implemented as Domain-Specific Languages (DSLs) [5]. Our approach clearly shows how model-driven engineering can contribute to DevOps by providing continuous testing and analysis processes for functional safety. Additionally, the use of DevOps techniques during the course of the development of this framework is also reported, where developers at different locations communicate with multiple academic and industrial users to continuously give and receive feedback. We report that using DevOps techniques in a model-driven engineering research project does not require a lot of additional effort and provides clear benefits.

The framework is built as part of the aSET project, which is briefly explained in Section II. In Section III, the approach is explained. In Section IV, we report on the use of DevOps techniques during the development of the framework. A conclusion is given in Section V.

II. The aSET Project

The framework presented in this paper is developed in the context of a project titled "Automated and Simulation-based Functional Safety Engineering Methodology" (aSET)[1]. aSET is a Interdisciplinary Collaborative Research (ICON) project, for which the goal is to transfer basic research (in this project, conducted by Flanders Make and the University of Antwerp) to industrial partners (in this project, Dana Belgium, Tenneco Automotive, Siemens Industry Software, HSPro). The project involves a strategic basic research part, in which common

[1]https://www.flandersmake.be/en/projects/aset

research questions are addressed to improve upon the state-of-the-art by the strategic research center. In the applied research part, the results of this research are then translated to the specific case of each of the industrial partners in separate work packages. This requires a close collaboration between academic and industry partners. In this paper, we focus on the results of the strategic basic research, as applied research results are confidential.

The industrial partners include manufacturers of mechatronic systems (Dana Belgium, Tenneco Automotive), simulation tool vendors (Siemens Industry Software) and engineering service providers (HSPro). All the industrial partners have the imperative of improving the reliability and efficiency of their functional safety engineering process. As mentioned, this process is hindered by the heterogeneous system artefacts and tools involved, rendering it difficult to create formal links between these artefacts. Also, as the functional concept of a system is commonly described textually or semi-formally, automated analysis is not possible.

Functional Safety Engineering Framework

Figure 1 shows the iterative development process (top part, "Design process" and "I/O"). Stages for the design process and (a selection of) involved engineering domains are shown, along with the input and output required for each stage. In a typical development process, iterations are made as engineers gain insight and discover issues, they may have to return to earlier steps, which increases the time and cost of the design process. Along with the development process, a safety analysis process is followed (middle, "Safety analysis"). There are multiple links between steps and artefacts of the development process, which are required by the ISO 26262 standard. These traceability links must be created and maintained through these multiple iterations and stages. Note that Figure 1 shows a simplified functional safety process, while in reality each shown safety related artefact typically consists of multiple models/files.

The main objective of the aSET project is to develop a framework to assist the functional safety engineering process. The framework aims for continuous integration of all artefacts throughout the functional safety engineering process, as well as the automation and early validation of requirements, to make the process less error-prone and to reduce the required design time and cost compared to the current (manual) state-of-the-practice. The framework consists of three main efforts, which are examined in the next section in the DevOps context:

- creating a Functional Safety Formal Model (FuSaFoMo) DSL containing the functional safety artefacts requested by ISO 26262 ("repository" at bottom of Figure 1);
- translating textual requirements into structured text and formal logic for verification ("requirements" at bottom of Figure 1);
- automation of Simulation-Aided Hazard Analysis and Risk Assessment (SAHARA) [6] (center of Figure 1).

At the time of writing, the aSET project is roughly halfway. A first prototype of the framework is ready, and is in use by industrial partners, such as for the industrial case study of an electronic differential lock.

III. DevOps for Functional Safety

This section will describe three different efforts considered in the aSET project for enabling continuous testing and validation in the design process for safety-critical automotive components. In particular, the project is focused on improving the iterative design process with model-based engineering techniques to ensure high-quality artefacts are available for analysis to engineers at all times. The three efforts are: a) creation of a formal safety model, b) improvements to the design process including formalization and verification of requirements, and c) improving the hazard analysis process through the use of simulation. Each effort will be discussed in terms of the intent, the successes, and the challenges encountered.

A. Formal Model for Safety

The main effort in the aSET project is the creation of a formal model language for describing safety-critical systems. This language, termed a *meta-model* in model-driven engineering [7], contains the types, multiplicities, and other constraints for writing a system design model. This Functional Safety Formal Model (FuSaFoMo) allows engineers to write a formal architectural description of the system in a similar fashion to the Analysis & Design Language (AADL) [8], and SysML block diagrams, but with precise semantics. The FuSaFoMo precisely targets the functional system domain, as it is built on concepts from the ISO 26262 standard. Currently, it is focused on the concept phase, as shown by the shaded part on the left side of Figure 1:

- item definition, which includes defining safety requirements and item boundary (i.e., scope of the functional safety analysis);
- HARA, in which hazards are identified and which results in safety goals to mitigate these hazards;
- safety-critical functions that are identified by HARA;
- functional safety concept, defining technical safety requirements, and which may introduces changes in the general functional requirements.

This effort to create a formal model is based on similar work in the literature focused on modeling safety domains [9], [10], [11]. The intention is to detect errors and inconsistencies earlier during the design, especially in the situation where multiple engineers are collaborating on the same system. The use of a single model to represent the architecture allows the engineers to avoid inconsistencies and have a high-quality artefact through the system design process.

The FuSaFoMo language and editor are built using the XText platform[2]. XText specializes in the creation of domain-specific languages (DSLs), which enable users to reason about a problem using the problem concepts, and not concepts from another domain such as code. Research has shown that this

[2]https://www.eclipse.org/Xtext/

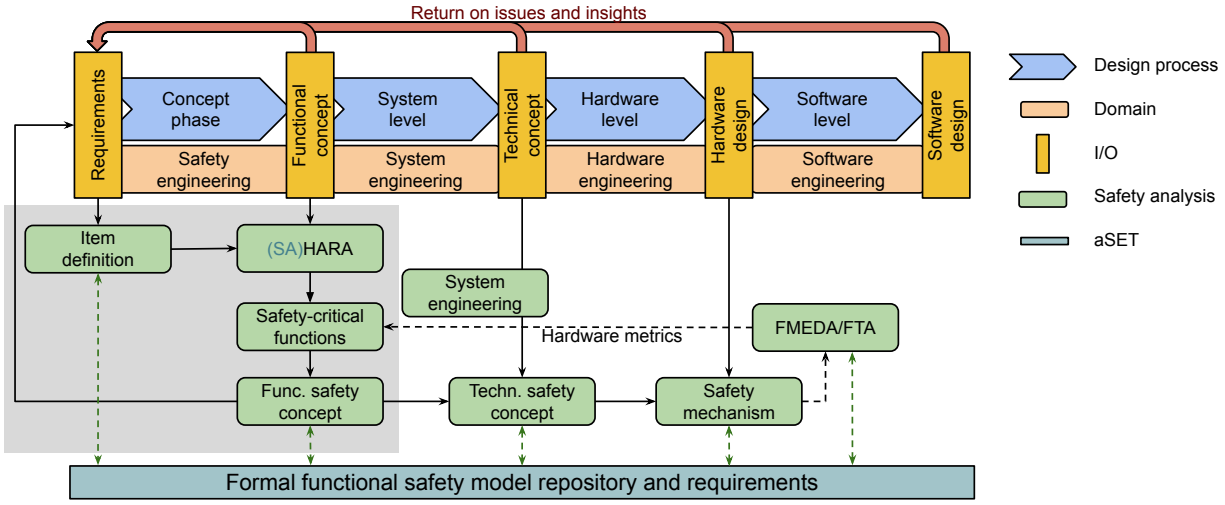

Fig. 1: Role of the framework within the design process.

can significantly increase productivity [5]. Additionally, XText provides rich support for the created language within the editor, including error-checking and auto-complete suggesting correctly-typed elements. This tightens the continuous testing loop to its maximum, as syntax errors in the model are caught immediately after they are made by an engineer, ensuring that artefacts are kept at high quality before being shared with other engineers.

A first version of the FuSaFoMo tool has been developed and is in use by the industrial partners. A number of inconsistencies have been detected by the tool, which were already present in the industrial partners' existing safety analysis components. These first results show how the tool can be used to support and partly automate the functional safety analysis process, and as an enabler to DevOps in safety-related systems.

Challenges: One key challenge is that the development process of the industry partners is flexible and differs between the partners and even within engineering teams. As an example, requirements may be made before or after the components they attach to in the design process. This means that the meta-model must avoid rigourous constraints such as requiring that a requirement always be connected to a component. However, this means that the ability of the meta-model to constrain the system to avoid errors is weakened. One possible solution to this challenge is to have a series of progressively more strict meta-models as the system becomes more developed, though this raises issues of models evolving over time to match the stricter meta-models.

A challenge in the effort to develop the FuSaFoMo itself is the requirement to align model versions belonging to multiple stakeholders. As the FuSaFoMo language is still in development, syntax changes mean that system models have to *evolve* to retain compatibility. This evolution is currently performed manually, though it is envisioned that (semi-) automatic model migrations could be made available when a new version of the FuSaFoMo language is created. An infrastructure for testing these migrations against sample system models could then lower the burden for users to switch FuSaFoMo versions.

B. Formalize the Design Process

Another effort in the aSET is to examine the design process of safety-critical components, and develop methods to formalize and enhance this process.

These stages of development and verification of the system is typically represented by a process diagram such as the typically used "V-cycle" [12], which has development of components down the left-side of the V, and integration and testing up the right-side of the V. As explained earlier, multiple loops exist in this process due to discovered issues. These loops back to an earlier step often brings high costs as designs must be modified and re-verified.

Such a design process could be made explicit through model-based techniques such as the Formalism Transformation Graph + Process Model [13]. This representation of the steps in the process and how they interact with model formalisms could enable (semi-)automatic enactment of steps including verification steps, and enhance the traceability of the process as required by the ISO 26262 standard.

The design of complicated systems can also be enhanced with the integration of a contract-based approach [14], [15], [16], [17] into the design process. The aSET project is exploring this direction with an effort on formalizing system requirements as structured contracts, which are then mapped into a temporal logic. An example of this structured language is seen in Figure 2, which offers operators to reason about the signals in a system, and *scopes* and *patterns* [18], [19] to define when the contract holds. The underlying Signal Temporal Logic (STL) can be automatically generated from these contracts and then verified on system models using toolboxes such as Breach [20]. Note that in these contracts,

```
Contract FR12{
        longname "Report to ESP"
        description " During normal operation and within
        t_SYSTEM_RESPONSE, the system must report to the ESP that
         the EDL is in one of the following two states:
        - Open; or
        - Closed.

        Unless the EDL is confirmed open, the system must report
        to the ESP that the EDL is closed."

        statements{
            Event enter_error_state := EnterErrorState == True,
            Event ReportState := EDLStateToESP from-table
                case SYSTEM_EDL_State != VERIFIED_EDL_State
                    : Set {EDL_State CLOSED}
                case SYSTEM_EDL_State == VERIFIED_EDL_State
                    : Set{ EDL_State OPENED, EDL_State CLOSED }
        }
        scope Before enter_error_state
        pattern Recurrence:
            ReportState occurs-repeatedly every 10 ms
    }
```

Fig. 2: Contract for verifying system behaviour.

operators are provided to possibly link signals to a FuSaFoMo model. This allows for the verification to be performed on simulations of the modeled component.

This contract also demonstrates an example of an inconsistency found during the formalization of the requirements. The description of the contract in Figure 2 mentions 'within `t_SYSTEM_RESPONSE`', which suggests that this contract refers to a required response within a timeout. However, as the figure shows, this requirement actually specifies a periodic task, where `t_SYSTEM_RESPONSE` is the period between the system requests.

Challenges: Currently in progress is the ability to have constant verification, where the STL can be verified against traces from the system during the writing of contracts. However, a key challenge is how to define the test conditions for each of these tests, especially as the models of the system may be under-specified during development.

C. Simulation-Aided Hazard and Risk Analysis (SAHARA)

The ISO 26262 standard focuses on the process for ensuring safety of automotive components, including defining the necessary steps to perform a Hazard And Risk Analysis (HARA). The aSET project has an objective to automate HARA to lower the time and cost of performing these HARAs on automotive components through fault injection and simulation techniques.

Following related work such as [21], one goal of aSET is to perform simulations on the behaviour of safety-critical components, under a variety of a) faults in the signals of the component (lack of signal, delayed signal, etc.), and b) road conditions (highway, rainy, crossing pedestrian, etc.). The outcome of these simulations could be a visualization so that the safety engineers performing the HARA have a greater appreciation of the scenario.

It is clear that the delay between a safety engineer wishing to visualize a scenario and obtaining that visualization should be as short as possible. This is due to the time- and person- cost inherent in the HARA discussion process. Therefore, efforts are being undertaken to a) provide a domain-specific language for scenario definition, and b) be able to rapidly produce these visualizations as the components and scenarios evolve. These artefacts would then be produced automatically for the safety engineers following changes in the modeled components or the scenario definitions.

Challenges: Various licensing and financial concerns have hampered the development of continuous testing procedures for performing these SAHARA simulations in a reproducible manner. For example, it is difficult to deploy the Simulink® tool in a container system for automated simulation due to these licensing issues, preventing full continuous integration and testing at this time.

IV. DevOps during aSET Development

This section will describe how DevOps techniques were used during the development of the aSET project itself. We focus on three directions which ensured constant, high quality releases to the industrial partners: a) continuous testing of artefacts, b) a culture of evaluation and feedback, and c) traceability of artefacts and discussions/decisions. While these activities may seem trivial to software developers familiar with DevOps principles, they are rarely seen in industrial and academic research projects such as aSET despite the clear benefits and minimal cost.

A. Continuous Testing

The aSET artefacts are contained in a GitLab[3] repository, including best practices documentation for users and developers, examples of tool use, and the artefacts for tool distribution. In particular, the Eclipse and XText plugins developed in the project for the FuSaFoMo and contract language are freshly built for each code commit, ensuring that there are no errors. There are also additional parsing checks, where a sample model is checked by the language parser. This ensures that syntax changes are explicitly thought about by the language designer, and that these tests can be properly communicated as examples for the industrial partners to change their models.

As previously mentioned, a challenge for this continuous testing is the setup of the containers and the testing process itself. Technical or licensing issues may hinder the process, as well as accessing proprietary data or models which is stored among the different partners in various tools. While this hinders complete regression testing, the solution is considered outside the scope of the aSET project.

The use of continuous testing and integration allows us to ensure that new features and fixes are automatically tested and released, so that the industrial users can easily obtain the latest version.

B. Feedback Culture

An important part of the aSET project is the culture of evaluation and feedback which has been fostered. Industrial and research partners communicate on the repository itself, utilizing issues to discuss key areas of concerns. Information has been also provided to the aSET partners on the Git

[3]https://gitlab.com/

concepts of tags, branches, and commits. This enables partners to accurately identify precisely where and when issues arose, and validate proposed features before integration into main development.

Challenges still to be overcome include motivating all relevant participants to be active in the feedback process, such as when key members are missing from a discussion. While this may be mainly an issue at the level of project management, there could be further technical assistance. For example, having fine-grained control on issues and commits to suggest their importance to various stakeholders. Another facet would be to move beyond the binary mention of a username in an issue, but instead to set users as blockers for the issue, such that it cannot process without their feedback.

C. Traceability

All aSET project discussions occur on the GitLab repository as comments on tracked *issues*. This allows issues to directly connect and evolve with the code and commits, as issues move from ideas to code to validation. This traceability is key for both the research and industrial partners. On the industrial side, it allows for knowledge dissemination, especially in an on-boarding process where new employees join the conversation. On the research and project managerial side, it allows for the tracking of key metrics and project status.

V. Conclusion

The goal of the Flanders Make aSET project is to deliver improvements to the design process of safety-related cyber-physical systems. This paper provides details to how DevOps principles such as continuous testing and continuous feedback are a) integrated into the design process of safety-critical systems, and b) used in the aSET project itself. Integration of DevOps principles into the design process can lower the time taken for costly functional safety design, by speeding up iteration time and catching errors earlier in the process.

As a final remark, we note that the project stakeholder's response to these principles has been positive, as DevOps gives confidence to our industrial partners to continuously use and evaluate our research and tools.

Acknowledgment

This work was partly funded by Flanders Make vzw, the strategic research centre for the Flemish manufacturing industry; and by the Flanders Make project aSET (grant no. HBC.2017.0389) of the Flanders Innovation and Entrepreneurship agency (VLAIO).

References

[1] R. Coppola and M. Morisio, "Connected car: technologies, issues, future trends," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 46, 2016.

[2] International Organization for Standardization, "ISO 26262: Road vehicles-functional safety," 2011.

[3] C. Becker, D. Arthur, and J. Brewer, "Functional safety assessment of a generic, conventional, hydraulic braking system with antilock brakes, traction control, and electronic stability control," National Highway Traffic Safety Administration, United States, Tech. Rep. DOT HS 812 574, Aug. 2018.

[4] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[5] J. Kärnä, J.-P. Tolvanen, and S. Kelly, "Evaluating the use of domain-specific modeling in practice," in *Proceedings of the Object-Oriented Programming, Systems, Languages and Applications workshop on Domain-Specific Modeling*, 2009.

[6] D. Szymanski, Y. Descas, K. Lehaen, and C. Mannaerts, "Simulation aided hazard analysis and risk assessment SAHARA," Conference presentation at IQPC Testing ADAS Self Driving Cars, Oberursel 2017.

[7] T. Kühne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[8] P. H. Feiler, B. A. Lewis, and S. Vestal, "The SAE architecture analysis & design language (AADL) a standard for engineering performance critical systems," in *Computer Aided Control System Design*. IEEE, 2006, pp. 1206–1211.

[9] K. Taguchi, "Meta modeling approach to safety standard for consumer devices," in *Seminar on Systems Assurance & Safety for Consumer Devices*, 2011.

[10] T. Frese, D. Hatebur, I. Côté, H.-J. Aryus, and M. Heisel, *Deriving Safety Requirements according to ISO 26262 for complex systems: A method applied in the automotive industry*. Springer Fachmedien Wiesbaden, 06 2017, pp. 211–221.

[11] de la Vara *et al.*, "Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel," *Information and software technology*, vol. 72, pp. 16–30, 2016.

[12] J. A. Estefan *et al.*, "Survey of model-based systems engineering methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[13] L. Lúcio, S. Mustafiz, J. Denil, H. Vangheluwe, and M. Jukss, "FTG+PM: An integrated framework for investigating model transformation chains," in *International System Design Languages Forum*. Springer, 2013, pp. 182–202.

[14] K. Vanherpen, "A contract-based approach for multi-viewpoint consistency in the concurrent design of cyber-physical systems," Ph.D. dissertation, University of Antwerp, 2018.

[15] A. L. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012. [Online]. Available: https://doi.org/10.3166/ejc.18.217-238

[16] P. Nuzzo *et al.*, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014. [Online]. Available: https://doi.org/10.1109/ACCESS.2013.2295764

[17] J. Westman, M. Nyberg, and M. Törngren, "Structuring safety requirements in ISO 26262 using contract theory," in *Computer Safety, Reliability, and Security - 32nd International Conference, SAFECOMP 2013, Toulouse, France, September 24-27, 2013. Proceedings*, ser. Lecture Notes in Computer Science, F. Bitsch, J. Guiochet, and M. Kaâniche, Eds., vol. 8153. Springer, 2013, pp. 166–177. [Online]. Available: https://doi.org/10.1007/978-3-642-40793-2_16

[18] M. Autili *et al.*, "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured English grammar," *IEEE Transactions on Software Engineering*, vol. 41, no. 7, pp. 620–638, 2015.

[19] B. Meyers, H. Vangheluwe, J. Denil, and R. Salay, "A framework for temporal verification support in domain-specific modelling," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[20] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.

[21] A. Duracz, H. Eriksson, F. A. Bartha, F. Xu, Y. Zeng, and W. Taha, "Using rigorous simulation to support ISO 26262 hazard analysis and risk assessment," in *Proceedings of High Performance Computing and Communications,*. IEEE, 2015, pp. 1093–1096. [Online]. Available: https://doi.org/10.1109/HPCC-CSS-ICESS.2015.296