A Model-Based Approach to Document Software Toolchains for Supporting a Safety Analysis

Stephan Baumgart*, Yin Chen†, Rasmus Hamrén‡, Sasikumar Punnekkat§
* Volvo Autonomous Solutions, Eskilstuna, Sweden. Mail: stephan.baumgart@volvo.com
†ABB, Västerås, Sweden. Email: yin.chen@se.abb.com
‡ Nordic Electronic Partner, Västerås, Sweden. Email: rasmus.hamren@nepartner.se
§ Mälardalen University, Västerås, Sweden. Email: sasikumar.punnekkat@mdh.se

***Abstract*—The increasing use of embedded systems to provide new functionality and customer experience requires developing the embedded systems carefully. As a new challenge, autonomous systems are developed to be working in a fleet to provide production workflows. Developing such a system-of-systems requires utilizing various software tools to manage the complexity. One task in developing safety-critical products, in general, is to analyze if the applied tools can introduce failures into the final product. Today's functional safety standards consider only single software tools for analysis. In our industrial work, we can observe a trend towards supporting product lines. A common configurable platform is developed to support a range of different products. Developing such a platform and supporting variability, a toolchain is created where software tools are glued together using scripts to support product lines and automatically generate compiled code. The current functional safety standards do not straight forward support this. This paper discusses how software tools need to support functional safety and show limitations by providing an industrial case. We provide a model-based approach to describe a toolchain and show its application to an industrial case. To analyze potential failures in the toolchain, we utilize the HAZOP method and show its application.**

***Index Terms*—Hazard Analysis, Toolchain, HAZOP, Safety, Model-based, System-of-Systems**

I. Introduction

Developing customer products requires ensuring and show that local regulations for different markets are fulfilled. To meet these regulations, specific standards have been developed to support developers in choosing the appropriate design and apply proven methods and processes. In this work, we solely focus on functional safety, which is described as the "absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems" [1]. Functional safety standards like the general standard IEC 61508 [2] or the domain-specific standards like ISO 26262 [1] for the automotive domain or ISO 13849 [3] and ISO 19014 [4] for the earth-moving machinery domain, describe processes and methods on how to develop the E/E system for safety-critical products. All possibilities that can lead to failure in the embedded system must be identified and possible mitigation is required to be implemented. Apart from requirements on the design of software and electronic hardware, the software tools applied in this process need to be analyzed. It is possible, that the correct programming code is leading to failures in the targeted embedded systems because of a malfunctioning compiler. The list of software tools used in an industrial development process can become long, integrating in-house-developed tools and third-party tools. Furthermore, these software tools are rarely applied in a stand-alone manner. Instead, these tools are connected to a complex toolchain by additional in-house-developed scripts for automation. The purpose of such automation can be to reduce the risk for human errors when creating the input for tools like compilers. Another reason to automate the toolchain is the high variability of the products to be supported. It is challenging to oversee all possible failures and dependencies and their impact on the generated software. The above-mentioned functional safety standards state requirements on a single tool and programming languages, but lack clarity on toolchains with a high degree of automation. There is a lack of guidance for practitioners on how to document the toolchain appropriately and analyze it for potential failures to meet the requirements from the relevant functional safety standards.

In this paper, we provide insights from our industrial work with the development of system-of-systems and provide practical guidance on how to document a toolchain. We utilize the hazard and operability study (HAZOP) [5] as an analysis method for studying the toolchain. We furthermore discuss our results concerning functional safety standard compliance.

This paper is structured as follows. We provide the background to our work in section II and describe an industrial case in section IV. Our approach to describe a toolchain to aid safety analysis is presented in section V. We discuss our findings and conclude our paper in section VI.

II. Background and Related Work

When developing safety-critical products, it is necessary to understand how failing product features can lead to critical situations with risk for health and safety. It needs to be understood how these features can fail and their causes. One way to introduce failure is a malfunctioning behavior of a software tool used to develop or verify the product. A software tool is a computer program used to assist the development, testing, or analysis. Examples of such software tools include compilers, automated code generators, and verification tools. It is necessary to analyze each applied tool in the development and verification processes for developing safety-critical products if malfunctioning of the tool or incorrect use of a tool can introduce faults in the final product.

Currently, tool qualification requirements are stated in different functional safety standards. However, these requirements are valid for single tools, not for a chain of tools. Furthermore, the product in this context is a single system and not a system-of-systems. A practicable qualification approach for a chain of tools is necessary and is beneficial for the complete process. In the following, we describe the requirements from relevant functional safety standards.

A. *Tool Qualification - Functional Safety Standards*

In general, different functional safety standards state requirements for tool qualification, such as ISO 26262 [1] or IEC 61058 [2]. Other functional safety standards like ISO 13849 [3], state rather vague requirements on tools focusing on proven in use. The argument "proven in use", requires that the tools are already used in previous development processes. This is not always possible since tools are also evolving or replaced.

1) IEC 61508: Generally, the functional safety standard IEC 61508 differentiates between if specific software support tools are applied either during the development and verification processes or during the product's run-time. Software tools applied during development and verification activities are referred to as software off-line support tools. Software on-line support tools "can directly influence the safety-related system during its run time" [2]. Such on-line tools shall be considered a software element of the safety-related system and shall be developed accordingly. In this work, we focus on software off-line support tools. To focus on the critical tools, each tool shall be classified by focusing on the possibility of introducing errors in the targeted code or by failing to detect errors.

- T1 (Support Tool): Category T1 covers all software tools with no direct or indirect impact on the code. Examples of T1 tools are text editors or requirements management tools.
- T2 (Verification Tool): In Category T2, all those tools are collected, which cannot change or affect the code of the safety-related system but can fail to identify defects in the code or the targeted E/E system. Typically, these can be verification tools that are failing to detect failures in the target.
- T3 (Development Tool): The category T3 covers those software tools which can generate or manipulate the code for the targeted safety-related system or an indirect impact on the code or its generation. Typical examples are compilers, which translate and optimize the source code.

A thorough analysis is required for all tools to identify if and how each tool impacts the safety-related system.

Each of the tools is seen as stand-alone, while the interaction between tools, automation of toolchains, or the impact on a system-of-systems is not considered. There are no requirements or guidelines on how a chain of tools shall be documented or analyzed in a structured way.

2) ISO 26262: ISO 26262 is providing a similar approach as IEC 61508 described above. The ISO 26262 utilizes two parameters to calculate a tool confidence level (TCL) for a software tool. The tool classification in ISO 26262 is based on the tool impact (TI) and tool error detection (TD).

Tool Impact (TI): The tool impact parameter is used to classify the software tools if they can have an impact on the safety-related system or not.

- TI1 contains all those software tools, where there is an argument that they are not in T2.
- TI2 contains all software tools directly or indirectly impacting the safety-related system or failing to detect failures.

Tool Error Detection (TD): The parameter TD covers the controllability dimension of risks related to tools, i.e., if the user of a tool has the possibility to detect a failure of the tool. TD1 contains those tools with a high probability for the user to detect a failure of the tool, while TD3 contains all those tools, with no possibility to identify a failure of a tool.

Both parameters are used to calculate the Tool Classification Level (TCL). The TCL is the highest (TLC3) for those tools, directly or indirectly contributing to the safety-related system's software and where the tool's potential failures are not detected. The same approach as described in IEC 61508 is used to state-specific requirements on tools of different criticality levels. No guidance is provided on how to deal with toolchains with a high degree of automation.

B. *Documenting Usage of Tools*

The functional safety standard ISO 26262 [1] is, for example, requiring knowledge about the intended purpose of a software tool, relevant inputs and outputs, and information about how the tools are used, including relevant constraints. Hillebrand et al. [6] have proposed a structured way of how to document tools. The authors utilize spreadsheets to list the tools, inputs, and outputs for each development phase in the V-Model. The authors do not consider the interaction of tools or automation of toolchains. Furthermore, the question of how to identify the critical system effects of the introduced faults is not discussed. Barner et al. [7] describe a toolchain for developing mixed-criticality systems. The purpose of the paper is not to discuss how to document a toolchain for conducting a safety analysis. Nonetheless, the toolchain is described as a process flow, which is different from the spreadsheet approach.

C. *System-of-Systems*

In our work, we focus, among others, on system-of-systems. ISO 26262 defines the term system as a "set of components or subsystems that relates at least a sensor, a controller, and an actuator with one another" [1]. The term system-of-systems is defined as "a set or arrangement of interdependent systems that are related or connected to provide a given capability" [8]. The standard ISO 21841 defines that system-of-systems consists of a "set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own" [9]. A constituent system in this context is an "independent system that forms part of a system-of-systems (SoS)" [9]. By integrating and connecting independent systems to system-of-systems, new dangers and risks may arise [10]. This means that even if integrating safety-certified systems into a system-of-systems, new hazards may result from the integration. Therefore, it is necessary to identify, analyze and capture those emergent hazards and review the integrated systems if undiscovered failures may lead to accidents with another constituent system.

III. Research Questions

In this work, we focus on complex toolchains for developing safety-critical products. The following research questions are answered as a contribution to this research.

RQ1 How can complex toolchains be documented using a model-based approach for enabling a safety analysis?

RQ2 How to identify failures introduced by the toolchain in the context of developing a system-of-systems?

IV. A Case Study from Industry

In this section, we present two case studies. The first case study covers the complex toolchain when developing embedded systems. In the second case, we describe the development process for a system-of-systems and explain the complex toolchain's specific challenges.

A. Case1: Development of Embedded Systems

With each new generation of products, the software tools used for developing the products are evolving. Furthermore, the industrial practice we study in our work is using a product line approach. This means that a configurable common platform is developed, used by several application projects, which develop the final products. Apart from the common software and common electronic hardware the platform provides, a toolchain is provided. These tools enable a configuration of the platform to the required needs.

To reduce the toolchain's complexity, it is important to group tools in process steps as shown in Figure 1. Each process step contains a set of connected tools, which are set up in a process. These processes can contain further steps, which may contain more details. Specifically, we can identify the following types of tools.

3rd Party Tools Third-party tools are all tools used in the toolchain provided by suppliers. Typically this can be tools like compilers, code checkers, and similar. For safety-critical tools, additional evidence may need to be provided by the supplier on how the software tool is developed and how risks related to the tool's usage have been considered during development.

In-house Tools Not all tools may be developed by suppliers. In-house developed tools can provide specific features to support the development processes.

In-house Developed Scripts Scripts are used to glue the tools together and provide the toolchain's features. By automating the inputs for all tools, the risk for user errors is reduced.s

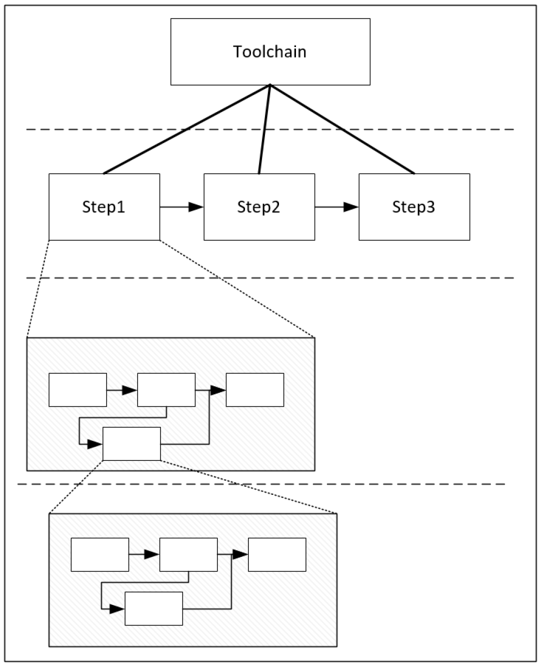


Fig. 1. Industrial case study - toolchain general

Characteristics of the industrial toolchain studied in this works:

- Variability: Support for product lines, the platform is configurable based on the needs of the targeted products
- Reducing Probability User Errors: Use of scripts to glue the tools together to reduce the risk for human errors when applying the tools.
- Evolution: Supporting, for example, new products or new electronic hardware makes it necessary to adjust parts of the platform

Information about third-party tools is available, including required inputs, target configuration, and outputs. Most interesting are the gluing scripts used for creating the toolchain.

Another aspect that needs to be considered is the possible changes that can happen over time:

- Changes in a script: A script is changed to improve the toolchain's provided functionality.
- Changes in software tool (internal, external): A software tool is changed, leading to changes in the toolchain.
- Changes in a process step: If a process step is changed, this may impact the complete toolchain.

Each of these changes may have an impact on the targeted product. Therefore, it is necessary to revisit the risk assessment of the toolchain as soon as changes are made. A structured method is necessary to support both the toolchain documentation and impact analysis in case of a change.

B. Case2: Development of System-of-Systems

We can observe a paradigm shift in the industry from developing single and human-operated machines towards developing autonomous vehicles used in a system-of-systems.

In Figure 2 we present a simplified view of the connected control systems. The autonomous vehicles, called TA15, are used to transport rocks or gravel in open-surface mines like quarries [11], [12], [13]. A central server coordinates the fleet of autonomous vehicles with a control system called Fleet Control. The TA15 machines are loaded with material using a human-operated Wheel Loader. Furthermore, a single TA15 can be controlled using a remote control. The communication between the server, the autonomous machines, and the Wheel Loader control system is realized through the wireless infrastructure.

When developing each control system by itself without considering integrating it into the system-of-systems, there is a risk of missing possible failures. It is important to identify which signals are critical and consider them when analyzing the toolchain.

V. Toolchain - Documentation and Analysis

In this section, we describe our method to specify the toolchain.

In Figure 3 we present the general process we propose to manage and document a software toolchain to conduct a safety analysis. This process contains three major parts. The first part is about where information about the software tools is stored. The second part focuses on how to document the toolchain. The last part is utilizing the information to feed a Hazop Analysis to identify hazards and risks related to the toolchain. One important input to the HAZOP Analysis comes from the product risk assessment. The critical system effects that shall be avoided are necessary input for finding critical software tools in the toolchain.

A. Tool Data from Database

The data about all types of tools are located in a separate database. The versions of 3rd party tools are fixed until a new version is accepted for the development process. In-house developed tools and the developed scripts are changing versions more often during the development progress. Therefore, it is necessary to provide versions of tools and input/output information to the modeling of the toolchain.

B. Documenting Toolchain

In this section, we specify what needs to be described

3rd Party Tools Third-party tools are all tools used in the toolchain provided by suppliers. Typically, this can be tools like compilers, code checkers, and similar.

In-house Tools Not all tools may be developed by suppliers. In-house developed tools and scripts may be used to provide required features to support the development processes.

Process Steps To reduce the complexity of the toolchain, it is important to group tools in process steps.

Process Files Process files are those files that are only used internally.

Input Files To analyze if the tool can introduce faults in the code, it is important to understand which files are used as input.

Output Files The same goes for the output files. It is important to understand what the tool is doing with the provided input files and which outputs are generated.

Unused Files Specifically 3rd party tools may provide a set of output files, which may not be used within the process. Nonetheless, such files may provide additional information, which can be used to verify the other outputs if required.

In Figure 4 we provide an example how to document the toolchain using SySML Block diagrams [14].

It is depicted which inputs are used in a certain tool or group of tools and generated. It is possible to describe a hierarchy of tools, which are refined on the next abstraction level. The process *Merge* shown in Figure 3 is not refined further, while the process *Generator Scripts* is further refined. Information about each tool's purpose can be added automatically by the direct connection to the tool database.

C. Hazard Analysis

We aim to analyze a complex process and utilize the Hazard, and Operability Studies (HAZOP) [5] as a method to identify critical tools in the toolchain. The main idea of the HAZOP analysis is to utilize guide words to structure the analysis efforts. The Hazop analysis as a method has its origin in the chemical industry [15], where complex processes need to be analyzed to identify critical scenarios. The method is used to identify the potential for system deviations from the intended operation.

We utilize the HAZOP method for analyzing the toolchain to 1) identify critical tools and 2) to identify possible mitigations. A mitigation for a specific risk related to a software tool could require additional verification activities to review a software tool's output. A wide range of guide words is listed in literature [5]. We chose to utilize the guide words "NO" and "WRONG".

- The guide word "NO" is used to find those failures, where a tool is not getting a required input or where an output file is not generated.
- The guide work "WRONG" is kept rather broad and may cover various types of failures.

In Figure 5 we provide an example for the use of the HAZOP analysis method for the toolchain. We use the following rows in the HAZOP table:

- Process Step: In this row, the main step is captured to raise awareness for the process currently studied. When many hundred scripts and software tools need to be analyzed, there is a risk of losing the overview.

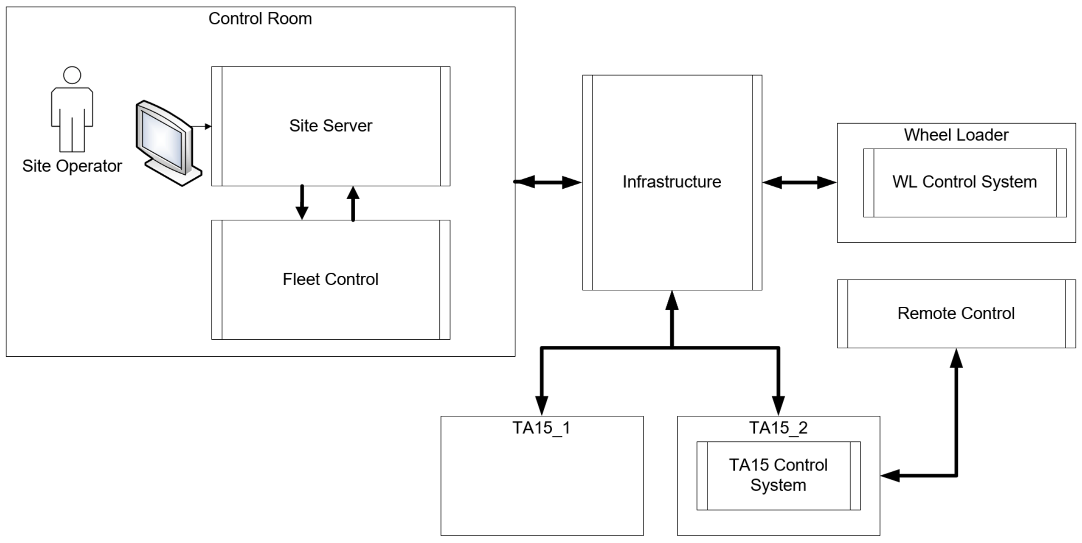


Fig. 2. Interaction of control systems in a system-of-systems

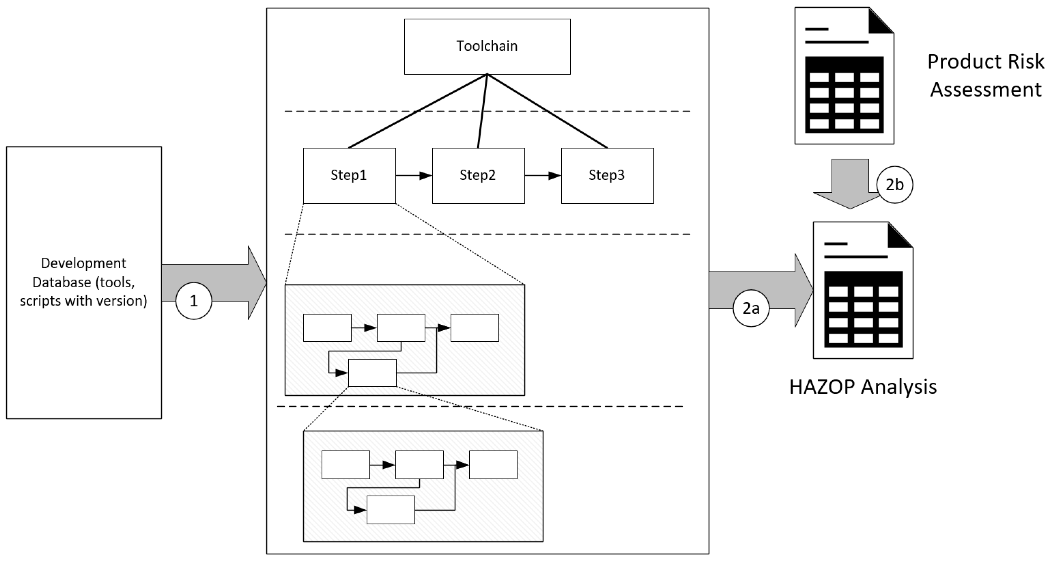


Fig. 3. Toolchain process

- Version: In this row, the versions of the tools to be analyzed are captured. The analysis is only relevant for a specific tool or script version. If a new version is provided or added, this tool's analysis needs to be repeated to ensure the new version is not introducing new failures.
- Tool/Script: The name of the tool or script is captured.
- Input/Output: Clarifying if the tool's input files are analyzed or the output files.
- File: Looking at the process as shown in Figure 4, the name of the specific file to be analyzed is captured.
- Guide word: In this row, the user guide words are listed.
- Cause: Here, we clarify what the possible cause for the failure is. This includes faults in a tool and a human error if, for example, a specific input needs to be provided by a user. The possible error types listed by Hillebrand et al. [6] are useful for guiding the analysis. The authors distinguish between user errors or tool internal errors.
- Consequence: In this row, the consequences of found deviations are listed.
- System Effect: It is important to relate the found software tool failures to the system level's possible effects.
- Recommended Action: During the brainstorming in the HAZOP meetings, possible actions can be identified, reaching from user educations to change requests.

The list of tools, versions, input, and output files can be generated from the SysML model shown in Figure 4. During

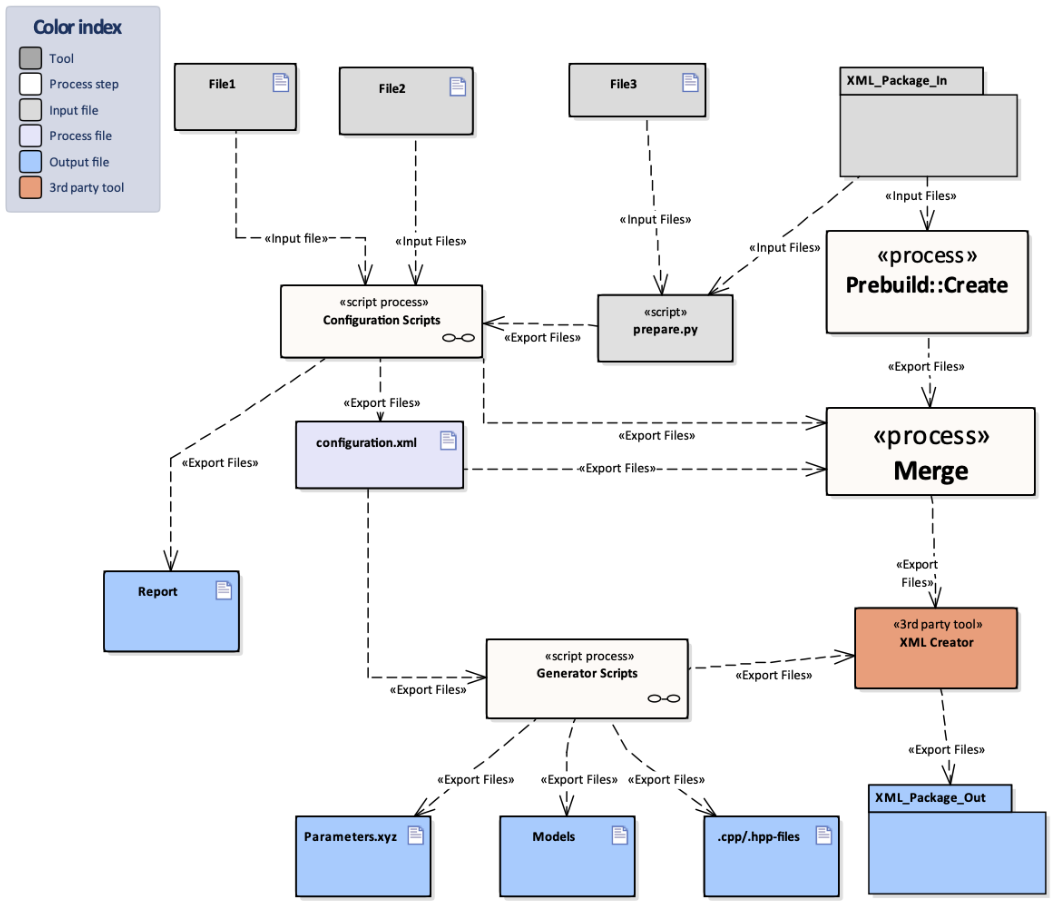
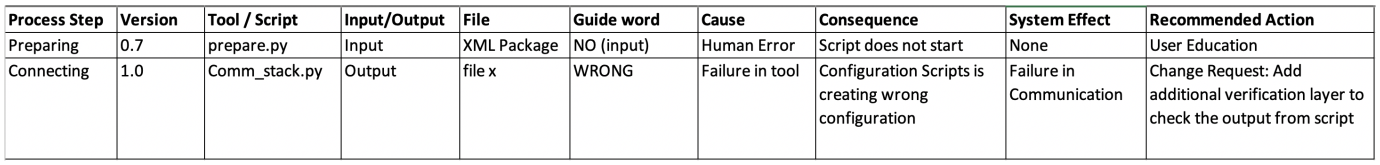


Fig. 4. Toolchain Specification - Example



Process Step	Version	Tool / Script	Input/Output	File	Guide word	Cause	Consequence	System Effect	Recommended Action
Preparing	0.7	prepare.py	Input	XML Package	NO (input)	Human Error	Script does not start	None	User Education
Connecting	1.0	Comm_stack.py	Output	file x	WRONG	Failure in tool	Configuration Scripts is creating wrong configuration	Failure in Communication	Change Request: Add additional verification layer to check the output from script

Fig. 5. HAZOP Toolchain - Example

the HAZOP meetings, each tool shall be evaluated. If tools or scripts are changing, the toolchain model will be updated accordingly. When exporting this data to the HAZOP table, we can highlight those tools in the HAZOP table that have changed or affected depending on how big impact the change has.

VI. Discussion and Conclusion

In this paper, we have discussed how software tools need to support functional safety. In this scope, we have discussed the limitations of existing concepts proposed in functional safety standards. The requirements in those standards focus on single and independent software tools. They do not guide how to document and analyze a toolchain. We have provided a case study of two industrial projects, one developing embedded systems and one developing a system-of-systems. We provide a model-based approach to describe a toolchain and show its application to an industrial case. To analyze potential failures in the toolchain, we utilize a HAZOP method and show its application. Since toolchains become more complex, we foresee the need to tool support for developers and safety engineers. Managing the number of tools manually and keeping track of changes, is time-consuming. More research is necessary to further study the impact of toolchain failures in the context of system-of-systems.

Acknowledgement

The authors acknowledge the funding support received for this research from Vinnova, the KKS-funded ITS-EASY Post Graduate School for Embedded Software and Systems.

References

[1] International Organization for Standardization, "ISO 26262:2018 - Road vehicles – Functional safety," 2018.

[2] International Electrotechnical Comission, "IEC 61508:2010 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems," 2010.

[3] International Organization for Standardization, "ISO 13849:2015 Safety of machinery - Safety related parts of control systems," 2015.

[4] ——, "ISO 19014:2018 Earth-moving machinery - Functional Safety," 2018.

[5] International Electronical Commission, "IEC 61882:2001 Hazard and operability studies (HAZOP studies) — Application guide," 2001.

[6] J. Hillebrand, P. Reichenpfader, I. Mandic, H. Siegl, and C. Peer, "Establishing confidence in the usage of software tools in context of ISO 26262," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, F. Flammini, S. Bologna, and V. Vittorini, Eds., vol. 6894. Springer Berlin / Heidelberg, 2011, pp. 257–269.

[7] S. Barner, A. Diewald, J. Migge, A. Syed, G. Fohler, M. Faugere, and D. G. Perez, "DREAMS Toolchain: Model-Driven Engineering of Mixed-Criticality Systems," in *Proceedings - ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS 2017*. Institute of Electrical and Electronics Engineers Inc., 10 2017, pp. 259–269.

[8] United States Department of Defense, "MIL-STD-882E," Washington, DC, USA, 2012.

[9] International Organization for Standardization, "ISO/IEC/IEEE 21841 Systems and software engineering — Taxonomy of systems of systems," 2019.

[10] P. J. Redmond, "A System of Systems Interface Hazard Analysis Technique," Master's thesis, 2007. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a467343.pdf

[11] S. Baumgart, J. Froberg, and S. Punnekkat, "Analyzing hazards in system-of-systems: Described in a quarry site automation context," in *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE, 4 2017, pp. 1–8.

[12] S. Baumgart, J. Fröberg, and S. Punnekkat, "A Process to Support Safety Analysis for a System-of-Systems," in *The 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020.

[13] Volvo Construction Equipment, "Electric Site Project," 2. [Online]. Available: https://www.volvoce.com/global/en/news-and-events/news-and-press-releases/2018/carbon-emissions-reduced-by-98-at-volvo-construction-equipment-and-skanskas-electric-site/

[14] Object Management Group, "SysML-Systems Modeling Language." [Online]. Available: https://sysml.org/

[15] D. Macdonald, *Practical Hazops, Trips and Alarms*, D. Macdonald and S. Mackay, Eds. Oxford: Newnes, 2004.