Tool Integration, from Tool to Tool Chain with ISO 26262

Fredrik Asplund, Matthias Biehl, Jad El-khoury, Daniel Frede, Martin Törngren
KTH Royal Institute of Technology

ABSTRACT

The use of innovative power sources in future cars has long-ranging implications on vehicle safety. We studied these implications in the context of the guidance on *software tool qualification* in the then current ISO 26262 draft, when building an urban concept vehicle to participate in the 2011 Shell Eco-Marathon. While the guidance on tool qualification is detailed, the guidance in regard to tools integrated into tool chains is limited. It only points out that the environment that tools execute in needs to be taken into consideration.

In this paper we clarify the implications of tool chains on tool qualification in the context of ISO 26262 by focusing on answering two questions; first, are there parts of the development environment related to tool integration that are likely to fall outside of tool qualification efforts as currently defined by ISO 26262; secondly, can we define *if*, and -if so- *how*, tool integration is affected by ensuring functional safety.

We conclude by identifying two areas related to tool integration that are likely to fall outside the tool qualification efforts (data integrity and process logic) and describing how different constraints imposed by ISO 26262 in relation to tool qualification conflict when tool integration is improved (improvements aimed at supporting completeness, consistency and the safety lifecycle vs. tool qualification cost).

We are able to make additional conclusions in relation to the *State of the Art* discussion on software tool qualification according to ISO 26262. First, reference tool chains and guidelines on which characteristics tool qualification should ensure for tool chains are needed to complement ISO 26262. Secondly, guidance on tool integration can be found in the completeness characteristic, the consistency characteristic and the ISO 26262 safety lifecycle process. Finally, qualification efforts should ideally target tool chains rather than individual tools.

INTRODUCTION

In this paper we clarify the implications of tool chains on tool qualification in the context of ISO 26262 [1]. We achieve this by focusing on answering two questions related to functional safety as specified by ISO 26262 and tool integration at a high level of abstraction; first, are there parts of the development environment related to tool integration that are likely to fall outside of tool qualification efforts as currently defined by ISO 26262; secondly, can we define *if*, and -if so- *how*, tool integration is affected by functional safety.

These questions enables a more detailed understanding of the required level of tool qualification in the context of ISO 26262, ensuring that important aspects of product development (including functional safety) are given appropriate consideration when weighed against each other. They also allow us to keep the focus on tool chains, while a discussion based on ISO 26262 is otherwise more likely to move into discussing individual tools (since this is what is dealt with in practice in ISO 26262). That we limit ourselves to ISO 26262 is, however, a choice based solely on the appropriateness of this standard to the domain of our case study. Tool integration is a complicated issue and other case studies could just as well give useful results based on other safety standards (such as DO-178 or IEC 61508).

To answer these questions we made use of the workflow depicted in *Figure 1*. We started with familiarizing ourselves with ISO 26262. This knowledge was then used to create a case study by building a tool chain compliant with ISO 26262 for a relevant development effort. The observations on this tool chain together with the experiences gained throughout the case study were related to a reference model for tool integration [2] to find functional safety characteristics to guide tool integration improvements. As a result a

number of suggestions of improvements to the tool chain could be made, which resulted in an improved tool chain both in regard to tool integration and ensuring functional safety. This improved tool chain was then analyzed and observations made regarding the impact of tool integration improvements on ensuring functional safety. The observations on both tool chains were finally related to the current *State of the Art* discussion on tool integration and *software tool qualification* according to ISO 26262, to allow us to both discuss the two questions detailed above and put them into a larger context.

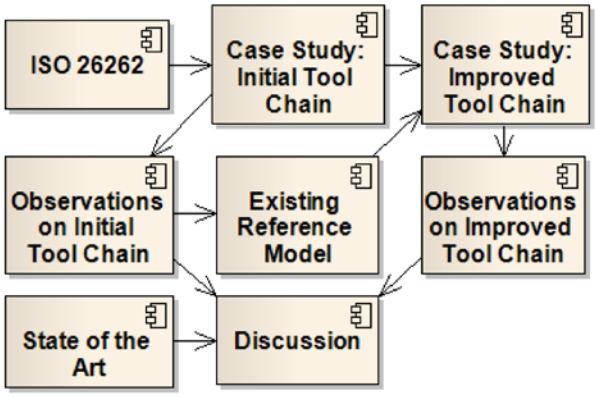

Figure 1 - Workflow

The most relevant part of ISO 26262 to this paper, the guidance on software tool qualification, is detailed in the next section. The case study and both the initial and improved tool chain are detailed in the third section. The reference model is presented by us in an earlier paper [2], but a summary of it can be found in the fourth section. The observations on the initial and improved tool chain are found in the fifth and sixth sections. In the seventh section we outline contemporary viewpoints, as represented by State of the Art research papers, on tool integration and software tool qualification according to ISO 26262. In the eighth section we bring together the fifth and sixth sections to discuss and answer our questions. In this section we also outline how our questions relate to the larger context described in the seventh section. In the final section we summarize our findings and point out the way forward to continue building upon these.

ISO 26262 AND SOFTWARE TOOL QUALIFICATION

ISO 26262 is an adaptation of IEC 61508 to comply with the needs specific to the application sector of Electrical and/or Electronic (E/E) systems within road vehicles [1]. It applies to all activities during the safety lifecycle of safety-related systems comprised of electrical, electronic, and software elements [1]. The aim of ISO 26262 is to ensure the functional safety of E/E systems.

Within this context ISO 26262 provides guidance on how to qualify software tools used when developing safety-related items [3]. This guidance is on a detailed level [4]. A tool qualification effort start with the gathering of information on the tool in question (identification, configuration, features, installation procedure, etc.), how it is used (use cases, the Automotive Safety Integrity Level (ASIL) of any product part it is used in the development of, etc.) and the context in which it is used (environment, user manual, etc.). Based on this information the tool qualifier (a vehicle or tool manufacturer, a supplier, a combination of these, etc.) is required to provide evidence of a sufficiently low possibility of safety requirement violations due to the use of the tool (regardless of whether the violation is caused by erroneous output from the tool, the tool failing to detect errors introduced by other tools, etc.). Determining what is the *correct* evidence starts with defining if a safety requirement can actually be violated. If this could be the case then the probability that this will not be prevented or detected is defined and the tool *classified* accordingly. The classification is considered in combination with the highest ASIL among the relevant parts under development. If a qualification effort is needed, then the evidence required by it could come from confidence from use, evaluation of how the tool was developed, validation of the tool itself and/or investigating if the tool has been developed according to a safety standard. A low probability of prevention or detection together with a high ASIL will imply a higher focus on the latter of these methods, and vice versa.

The described procedure is open-ended, with room for interpretation and further guidance through analysis and reference qualifications. One of the factors of which the content is open to interpretation is the *development environment* that a tool executes in, i.e. all the other tools and any supporting software. An important aspect of a development environment is that the *development process* it supports will define orderings of the tools it contains, i.e. *tool chains*, which the development of a product will transit through. We define *tool integration* as supporting the development process in correctly moving from one tool to another in a tool chain. Tool

integration may play a large part in tool qualification efforts, since it has the potential to define much of the context in which any specific tool is used.

THE CASE STUDY

In the Spiros IV project[1] staff from several departments at KTH cooperated to build an Urban Concept vehicle to compete in the 2011 Shell Eco-Marathon. As responsible for the safety of the vehicle we set up an initial Hazard and Operability Study (HAZOP) [5] with representatives from all the development areas. During the discussions we identified that the Hydrogen Powertrain Safety System (HPSS) was not safe enough even though it was acceptable according to the rules of the competition. While the rules ensured safety in nominal circumstances, the safety of the HPSS in other than nominal circumstances was not ensured. To remedy the situation we decided to develop a Hydrogen Leakage Warning System (HLWS) according to the requirements for ASIL D development defined in ISO 26262. ISO 26262 is intended for E/E systems that are installed in series production passenger cars and does not address unique E/E systems in special purpose vehicles. Therefore we took steps to ensure that our development was sufficiently close to the spirit of ISO 26262 to allow it to be used as a case study for studying functional safety in relation to tool integration. The foremost of these steps were to enlist an external reviewer[2] with a long history of applying the ISO 26262 drafts published thus far. Other steps were to carefully review and analyze parts of ISO 26262 that we could not fulfill (mostly requirements on destructive testing) and use pessimistic estimates when information was not conclusive.

The content of the case study is the set up of an initial tool chain to support the safety lifecycle during the HLWS development and subsequent efforts to improve the tool integration. The initial tool chain was set up to contain little tool integration. This made it easier to identify the minimal impact of ISO 26262 and distinguish the minimal tool integration required. The improvements to the tool integration were made after the Spiros IV project, to allow drawing on both experiences from the whole HLWS development lifecycle and observations on the initial tool chain to identify changes that would help ensure functional safety.

In the initial tool chain Microsoft Word and Excel were used for work products stipulated by ISO 26262 for the Supporting Processes and in the Management of Functional Safety, Concept, System Level, Hardware Level and Production and Operation Phases[3]. National Instruments (NI) Multisim and CadSoft EAGLE Layout Editor were used for drawing the Detailed Design in the Hardware Level Phase. No software was developed for the HLWS. The Git Version Control System (Git VCS) was chosen to ensure the requirements by ISO 26262 on controlled distribution of external documents, version handling, document identification and controlled approving and obsoleting of documents (ISO 26262 requires configuration management based on well-established standards [6]. We chose to use ISO TS 16949 for guidance [7].). The Windows XP and Ubuntu 10.04 LTS Operating Systems were used as development platforms. The input to and output from each tool was subject to review during manual transfer of data between tools. This increased the possibility to catch errors before they could lead to violations of a safety requirement.

The part of the initial tool chain relating to the Hardware Level of product development is depicted in *Figure 2* and *3*. The Hardware Level is representative for the whole case study, since most of the tool integration took place at this level. The corresponding part of the improved tool chain is presented in *Figure 4*. The changes suggested are the use of application start and terminal control services (by changing from EAGLE Layout Editor to NI Ultiboard and using the LabVIEW Multisim Connectivity Toolkit), merging data interoperability sets[4] through links and transformations (using a linking tool and functionality built into the NI tool suite) and evaluating the hardware architectural metrics with an analysis tool.

The format of *Figure 2*, *3* and *4* depicts the fragmentation of the aspects of tool integration and the dependencies between them that are supported by the development environment [2]. Each box marks an instance of a tool integration aspect, visualizing if and how these are shared across tools and development phases. Both the aspects and their possible dependencies are described in the next chapter.

---

[1] http://www.spiroscar.se

[2] Assistant Professor Chen De-Jiu, KTH, chen@md.kth.se (independent from Spiros IV, the organization responsible for the considered work products regarding management, resources and release authority)

[3] These are terms from the ISO 26262 reference process model for the different phases of product development.

[4] A data interoperability set is a collection of syntax and semantic pairs that are related by links or transformations.

THE REFERENCE MODEL

To analyze the case study we related it to a reference model for tool integration in which non-functional properties (such as functional safety) are of central importance [2]. To prepare for the subsequent chapters this reference model is summarized here.

The structure for reasoning about tool integration is supported by the reference model by dividing tool integration into *control*, *data*, *platform*, *presentation* and *process integration aspects* and defining *dependencies* and *metrics* for these.

- Control integration is the degree to which tools can issue commands and notifications correctly to one another. The metric of control integration is the number of unique services in a system. Control integration is dependent on platform integration, since services are accessed via a common environment.
- Data integration is the degree to which tools can, in a meaningful way, access the complete data set available within the system of interest. The metric of data integration is the number of data interoperability sets in a system. Data integration is dependent on control and platform integration, since access of data is dependent on services and a common environment.
- Platform integration is the degree to which tools share a common environment. The metric of platform integration are the number of distinct platforms used in the system.
- Presentation integration is the degree to which tools, in different contexts, can set up user interaction so that the tool input/output is perceived correctly. The metric of presentation integration is the level to which users are forced to view or manipulate data in views they are not used to interact with. Presentation integration is dependent on control, data and platform integration, since these limits the ways of presenting input/output to users.
- Process integration is the degree to which interaction with tools can be supervised. The metric of process integration is the number of services that allow the end-user to know the state of the system when they signal completion. Process integration is dependent on control and data integration, since process awareness is propagated by the former and process state is decided by the latter.

For a given, static setup of a tool chain, non-functional properties provide limitations and are supported by specific characteristics of the tool chain. For improving tool chains, the metrics provide guidance on how to improve tool integration, but these improvements can also have the consequence of affecting non-functional properties positively or negatively. In the two subsequent chapters we will relate this reference model to our case study to make observations on how functional safety as specified by ISO 26262 relates to tool integration. Non-functional properties may be interdependent, even conflicting, but in this paper we limit ourselves as much as possible to functional safety to avoid unnecessary complexity. Further work will analyze in detail how functional safety relates to other non-functional properties.

OBSERVATIONS ON THE INITIAL TOOL CHAIN SETUP

As mentioned in the previous chapter, when using the reference model in question to deal with a static tool chain setup, the role of non-functional properties is twofold. First, they provide limitations on what is acceptable. Secondly, they are supported by characteristics of the tool chain.

LIMITATIONS

The guidelines for qualifying tools can be extended to the whole development environment, but this implies a large effort (envision qualifying the platform of our case study, i.e. the combination of Windows XP and Ubuntu 10.04 LTS). In our minimal setup and in relation to control, data and platform integration, the only tool integration required was the transfer of data between the implementer of a part of the development process and the reviewer of said part. Applying the ISO 26262 tool qualification guidelines on this tool integration implies ensuring the integrity[5] of data as it is passed around between different platforms and services. In our case study this was handled by the Git VCS, which ensured that a committed file could not change without detection. One could avoid this integration in a monolithic system, but that would be the odd case rather than the norm. While ISO 26262 strictly speaking does not *require* any control, data or platform integration, it will impose the limitation that data integrity needs to be ensured by these integration aspects in most development environments.

[5] Integrity in the broader sense, i.e. in regard to the reliability that the data reflects the current state of the development.

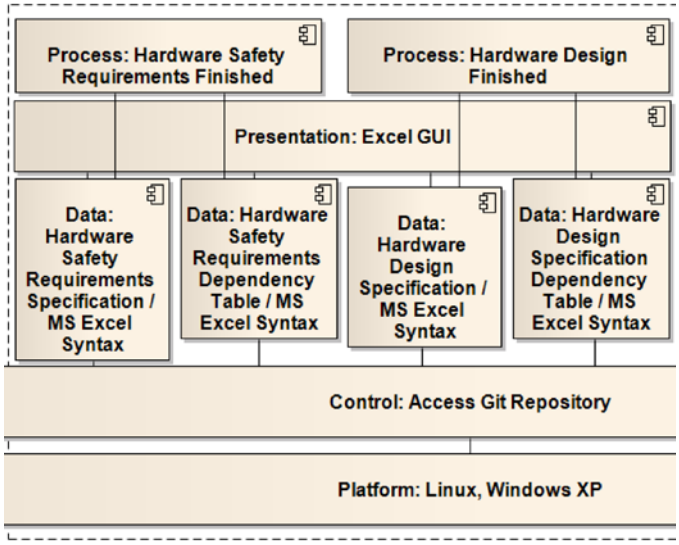

Figure 2 - The safety-related parts of the Spiros IV tool chain for the Hardware Level, Part 1.

The only part of ISO 26262 that is relevant for presentation integration is the stipulated configuration management that requires all software to be able to visualize the version and status of work products and documents [6]. In our case study we relied on support from Git VCS to store files in different folders and branches to achieve a part of this for the whole tool chain. However, this visualization could have been achieved in a domain-specific way within all the used tools (using text labels). We could not identify any presentation integration *required* by ISO 26262, nor any limitations on presentation integration due to functional safety. However, given that users are more likely to make mistakes when forced to view or manipulate data in views they are not used to interact with (due to an erroneous understanding of their environment), there is likely to exist at least some best practices to adopt (a relevant discussion about safety issues related to users and automation can for instance be found in [8]).

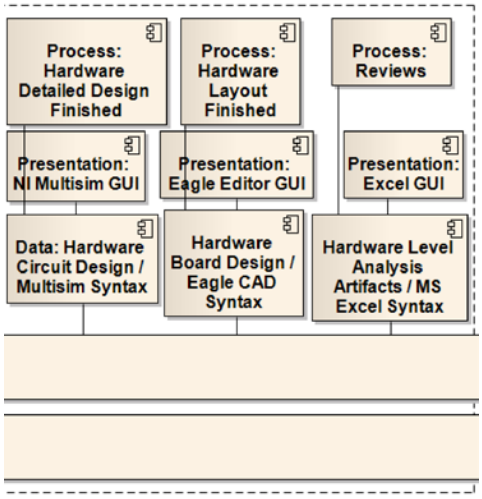

Figure 3 - The safety-related parts of the Spiros IV tool chain for the Hardware Level, Part 2.

ISO 26262 defines a development process to ensure functional safety, so all the used orderings of tools need to comply with this process. This was ensured in our case study by not applying any automation to the moves between process activities and requiring reviews between them. We could not identify any process integration *required* by ISO 26262, nor any limitations on process integration due to functional safety (the discussion mentioned in the previous paragraph is also relevant here [8]).

SUPPORT

On a high level of abstraction ISO 26262 stipulates a number of development work products (the safety plan, the safety case, etc.), certain characteristics to be proven for specific development work products (the hardware architectural metrics, the supporting processes, etc.) and continuous reviews requiring *consistency*[6], *completeness*[7] and *correctness*[8] between different process activities (functional safety requirements to technical safety requirements, technical safety requirements to system design, etc.) [1]. The requirements for consistency (when the work products in question belong to different tools) and completeness are related to tool integration, while the others are related to tools (the hardware architectural metrics can be proven using analysis tools, etc.). There could be a need for tool integration in relation to the work products, characteristics or the requirement for correctness (the data needed to calculate the hardware architectural metrics could for instance have to be gathered from several sources and transformed into a specific syntax), but these are issues that can be related to specific use cases and tools. The requirements on consistency and completeness by contrast impact a large continuous portion of the development environment and are therefore strong markers of an implicit requirement for focusing on the development environment as part of a system (rather than just focusing on specific, well defined characteristics of well specified work products or processes).

OBSERVATIONS ON AN IMPROVED DEVELOPMENT ENVIRONMENT

The improvements (described in "The Case Study" section) that we deduced as beneficial for functional safety, mainly due to our observations on how functional safety was supported by the initial tool chain, are captured in the improved tool chain. The reference model metrics, shown in *Table 1*, were used to identify whether tool integration was improved by these changes and allowed us to characterize the impact of functional safety on tool integration.

Table 1 - Integration Aspect Metrics

Integration Aspect	Metric – Original Tool Chain	Metric – Improved Tool Chain
Control Integration	1	3 (Increased tool integration)
Data Integration	9	4 (Increased tool integration)
Platform Integration	1	1
Presentation Integration	0	1 (Decreased tool integration)
Process Integration	0	0

The need to ensure completeness and consistency between work products at several process activities was initially ensured by dependency tables and manual reviews between each process activity. Including more tool integration to support automation is one way of guaranteeing that the claims for completeness and consistency are valid.

[6] We define consistency as the requirement that no work product should contradict another, for instance by requiring an end product to exhibit two mutually exclusive characteristics.

[7] We define completeness as the requirement that all process activities should cover the same requirements, albeit be it at different levels of abstraction.

[8] We define correctness as the requirement that all work products should be logically correct, i.e. not contradict physical laws or any logical requirement used during development.

- The use of additional services to automate the transition between certain process activities (starting an executable from inside Multisim and using the LabVIEW Multisim Connectivity Toolkit) was suggested, which would increase the level of control integration. This would decrease the likelihood and efficiency of manual reviews, since the users are being distanced from the actual development process (i.e. their knowledge of what is actually happening is likely to decrease). The result is a decrease in confidence of catching errors during tool qualification. However, the misuse of services (even combinations of them) are already required to be handled during tool qualification, so there is no additional qualification effort *required* on the actual software used to support the use of services. That said, there could be an added qualification effort linked to the use of additional services due to more, larger or more complex use cases. Qualifying for instance a message server could then be beneficial by supporting confidence during tool qualification.
- The increased automation also provides the possibility to, or even requires one to, merge data interoperability sets by links or transformations (we suggested the use of a linking tool, but links as part of the data can also be envisioned). While the reviews discussed above would lose significance, ISO 26262 still stipulates reviews of data to ensure completeness and consistency in several of the phases of the safety cycle [1]. This means that no additional requirements would be added to tool qualification, but the increased data integration would entail the need to qualify the logic of the data interoperability set merging (no dangling references, correct transformation semantics, etc.). This would help guarantee the basis for reviews.
- Supporting reviews of completeness and consistency by increased automation would mean to access work products pertaining to different development phases at the same time (albeit that it could be in a separate tool and only in relation to two adjacent phases at a time). The increased requirement on presentation integration to present different abstractions together would have to be handled when qualifying the relevant tools and may require working with the limitations set by control, data and platform integration.
- The ISO 26262 safety lifecycle does not lend itself easily to process orchestration[9] due to the high focus on reviews, walkthroughs and inspections, but we could see instances when it would be appropriate (for instance when evaluating hardware architectural metrics with an analysis tool). The safety lifecycle does, however, give detailed metrics for evaluating the state of the data in relation to the different process activities, which would be helpful for process integration even when the process is being driven by manual decisions. The impact on process integration is twofold; it implies that any process orchestration needs to be qualified to only allow workflows defined as correct by ISO 26262; it also provides input to when a process activity can be seen as complete.

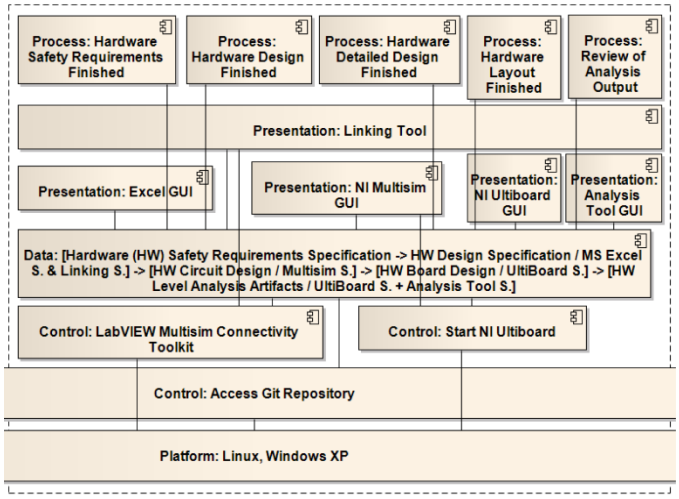

Figure 4 - Improving the safety-related parts of the Spiros IV tool chain for the Hardware Level.

In the next two chapters we will relate the observations we have made to both the current discussion on ISO 26262 and the questions detailed at the start of this paper.

[9] Coordination of services and events to create a composite process (usually implying the existence of a single coordinating force).

RELATED WORK

Tool integration has been an active field of research for several decades, but there are still many unanswered questions regarding how to reason around this topic at an abstract level. The focus has traditionally been on technical problems [9] even though these are only the end result of a strive to handle complexity, remove the likelihood of inconsistencies, provide tracing, etc., to support the primary goals of ensuring safety, cost-efficiency, etc. This means that the answers provided by the abundant and diverse amount of reference models based on checklists [10] or measurable characteristics [11] are not always helpful when analyzing a development environment. Attempts to capture the impact of tool integration on non-functional properties and vice versa are uncommon with the only major contribution we have been able to find concerning cost in the form of a software cost estimation model [12]. Trying to relate functional safety to tool integration at an abstract level seems to be an unexplored field of study.

The contributions regarding ISO 26262 related to our paper concern software tool qualification. The prevailing idea is for tool vendors to provide tool qualification based on reference workflows [3, 4, 13]. This would provide guidance to tool qualifiers or even allow them to limit their efforts to qualifying only the differences between the reference workflows and their own. How to deal with tool chains and, implicitly, tool integration is however the subject of different opinions. One view is that one should first investigate if tools are safeguarded by any subsequent tool in the tool chains they are part of, thereby allowing the qualification effort to be spent only on the critical use cases related to the first and last tools [13]. This would allow tool qualifiers to limit very large total qualification efforts to “reasonable” sizes. Another view is that relying on a subsequent tool as a safeguard will lower the confidence in the tool chain as a whole [4]. This would imply qualifying each tool chain as a whole or introducing metrics for qualification when combining tools.

DISCUSSION

We explore two questions in this paper, the first regarding whether there are parts of the development environment related to tool integration that are likely to fall outside the tool qualification efforts as currently defined by ISO 26262. Due to the fact that ISO 26262 does not stipulate any types of tool or work product formats, it is not surprising that the stipulated way of tool qualification is primed to cover an extensive part of any development environment. No tool can be qualified without taking into consideration the context in which it executes. ISO 26262 does, however, imply that scrutinizing each *tool* thoroughly enough can eventually ensure functional safety. Tool integration deals with the interaction between tools, which is not always possible to overview from inside a tool. The state of the different work products in a development environment can be logically correct without being correct in regard to the development process, for instance when characteristics of a product at one product activity contradict those of the next phase, parts of a tool chain are possible to arbitrarily ignore, the wrong version of data is handed over to the next process activity, etc. The term *software tool* can undoubtedly be interpreted in a more generic sense to account for this when applying ISO 26262 on a development environment, but several sources of pressure to interpret the term more narrowly is apparent (economic, performance related, etc.). We observed two areas that are easy to overlook, since while they are directly related to tool integration they may only be indirectly related to distinct tools. These areas were data integrity and the adherence to workflows conforming to ISO 26262. Of these the latter was not directly related to any tool in our case study. The former was directly related to a tool (Git VCS), but since this tool had a supportive role rather than being directly involved in the development process it is also easy to overlook during tool qualification.

That issues such as cost are conflicting with the control of a safety-related process is not uncommon, but to ensure that this does not compromise safety the necessary constraints need to be clearly understood. That ISO 26262 leaves the question of which tools that should be qualified to the tool qualifier is in itself not problematic, but it becomes so in combination with the lack of guidelines on which characteristics tool qualification should ensure for tool chains. The result is a vague control of which software should be qualified. This may imperil functional safety goals if distributed or indistinct software is used to handle important areas, since no guidelines are in place to prompt the tool qualifier to analyze this software thoroughly. In practice, taking the two identified areas into consideration in reference workflows for certification purposes is straightforward. However, other areas such as these may exist, since we did not perform any exhaustive search. Even worse for those interested in qualifying separate tools, it casts doubts on the sufficiency or even the possibility to ensure the correct functioning of each tool in isolation.

The second question is *if*, and -if so- *how*, tool integration is affected by the need to ensure functional safety in the context of ISO 26262. The result from our case study is the following:

- Improving control integration is desirable, but will imply a higher qualification cost both by decreasing confidence and expanding use cases.
- Improving data integration is desirable, but implies an increasing need to qualify data interoperability set merging.
- Improving process integration is desirable in regard to process control, but implies an increasing need to qualify the adherence to correct workflows.

These improvements are desirable when they support three constraints which ISO 26262 stipulates to ensure functional safety; the completeness characteristic, the consistency characteristic and the ISO 26262 safety lifecycle process. The control of these three constraints can be supported by tool integration, while the control of the constraint that tools should be qualified is aggravated by tool integration. Thus tool integration creates a tension between these four constraints, which we can observe and analyze. Ensuring functional safety in regard to tool integration becomes a problem of ensuring the control of all these constraints together and this should guide tool integration when safety is a concern. This implication also impacts the discussion on using tools in the end of tool chains to safeguard the other tools in large development environments. If the completeness and consistency characteristics are constraints by themselves, then qualification efforts must enforce them throughout the safety lifecycle. In that case the classification of tools throughout tool chains will be impacted, something which is only augmented by the introduction of additional tool integration mechanisms as a development environment scales up. This will decrease the effectiveness of a strategy that tries to concentrate the efforts to the end of tool chains.

SUMMARY/CONCLUSIONS

We have identified two areas related to tool integration that are likely to fall outside the tool qualification efforts, namely data integrity and process logic. We have also identified that control, data and process integration improvements aimed at supporting completeness, consistency and the safety lifecycle process are desirable in regard to functional safety, but imply increased qualification costs.

We are also able to make additional conclusions in relation to the State of the Art discussion related to ISO 26262. First, reference tool chains and guidelines on which characteristics tool qualification should ensure for tool chains are needed to complement ISO 26262. Secondly, guidance on tool integration can be found in the completeness characteristic, the consistency characteristic and the ISO 26262 safety lifecycle process. Finally, qualification efforts should ideally target tool chains rather than individual tools.

Having identified areas related to tool integration and characteristics of tool chains that are important in the context of safety as defined by ISO 26262 the next step will be to expand on and generalize these findings. Especially two questions are highlighted by our findings:

- Arbitrarily increasing tool integration seems to mainly impact safety by placing additional burden on software qualification, since the complexity of the overall system increases.
    - As a development environment scales up, is the software qualification method stipulated by ISO 26262 the best way forward?
    - Can the implied increase in software qualification efforts be mitigated by special treatment of certain areas or the by ensuring that the used tool chain has some specific characteristics?

To answer these questions we will assess our findings by analyzing another safety-related case study on tool integration using a generic hazard analysis technique (such as STPA [8]).

REFERENCES

1. ISO/FDIS 26262:2010 - Road vehicles - Functional safety, International Organization for Standardization Std.
2. F. Asplund, M. Biehl, J. El-Khoury, and M. Törngren, “Tool integration beyond Wasserman,” in Advanced Information Systems Engineering Workshops: CAiSE 2011 International Workshops, London, UK, June 20-24, 2011, Proceedings, 2011, pp. 270–281.
3. M. Conrad, G. Sandmann, and P. Munier. Software tool qualification according to ISO 26262, SAE 2011 world congress & exhibition, april 2011, Detroit, MI, USA.
4. M. Conrad, P. Munier, and F. Rauch, “Qualifying software tools according to ISO 26262,” in Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VI, 2010, pp. 117–128.
5. J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, “Hazard and operability (HAZOP) analysis. a literature review,” Journal of Hazardous Materials, vol. 173, pp. 19–32, 2010.
6. ISO/FDIS 26262-8:2010 - Road vehicles - Functional safety - Part 8: Supporting processes, International Organization for Standardization Std.
7. ISO/TS 16949 - Quality management systems - Particular requirements for the application of ISO 9001:2000 for automotive production and relevant service part organizations, International Organization for Standardization Std.
8. N. G. Leveson, Engineering a Safer World, Systems Thinking Applied to Safety (Draft). MIT Press, 2011.
9. M. Wicks and R. Dewar, “A new research agenda for tool integration,” The Journal of Systems and Software, vol. 80, pp. 1569–1585, September 2007.
10. M. V. Zelkowitz, “Use of an environment classification model,” in ICSE ’93: Proceedings of the 15th international conference on Software Engineering, 1993.

11. S. Izza, “Integration of industrial information systems: from syntactic to semantic integration approaches,” Enterprise Information Systems, vol. 3, pp. 1–57, February 2009.
12. J. Baik, B. Boehm, and B. M. Steece, “Disaggregating and calibrating the CASE tool variable in COCOMO II,” IEEE Transactions on Software Engineering, vol. 28, pp. 1009–1022, 2002.
13. R. Hamann, S. Kriso, K. Williams, J. Klarmann, and J. Sauler. ISO 26262 release just ahead - remaining problems and proposals for solutions, SAE 2011 world congress & exhibition, april 2011, Detroit, MI, USA.

CONTACT INFORMATION

KTH, Brinellvägen 83, 10044 Stockholm, Sweden

{fasplund, biehl, jad, frede, martint}@kth.se

ACKNOWLEDGMENTS

We thank all participants of the ARTEMIS iFEST project, who have given us continuous access to an additional breadth of expertise on and experience of software engineering in relation to the lifecycle of embedded systems.

DEFINITIONS/ABBREVIATIONS

ASIL	Automotive Safety Integrity Level
E/E	Electrical / Electronic
HAZOP	Hazard and Operability Study
HLWS	Hydrogen Leakage Warning System
HPSS	Hydrogen Powertrain Safety System
NI	National Instruments