<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper published in *Safety Science*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Asplund, F. (2015)

The future of software tool chain safety qualification.

Safety Science, 74: 37-43

<http://dx.doi.org/10.1016/j.ssci.2014.11.023>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-163963>

The Future of Software Tool Chain Safety Qualification

Fredrik Asplund^{a,1,*}

^a*KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Brinellvägen 83, 10044 Stockholm, Sweden*

Abstract

High profile systemic safety standards for Cyber-Physical Systems (CPS) development within the *transportation domain* have commonalities with regard to their view of the safety-related implications of tool usage. Their guidelines on *tool qualification* favour a bottom-up approach in which tools are dealt with in isolation and mostly if they may directly introduce faults into end products. This guidance may ignore risk introduced by the integration of software tools, especially if these risks are related to low levels of automation - such as process notifications and improper graphical user interfaces.

This paper presents a study that ties weaknesses in support environments to software faults. Based on the observed weaknesses guidelines for a top-down software tool chain qualification are suggested for inclusion in the next generation of safety standards. This has implications not only for the surveyed standards in the transportation domain, but also for other standards for safety-critical CPS development that do not include a broader view on risks related to tool usage.

Furthermore, given the type of omission identified in the surveyed standards, it is suggested that researchers interested in the safety-related implications of tool integration should approach organizational research in search of possibilities to set up theory triangulation studies.

Keywords: Automation, Certification, Tool Qualification, Support Environments, Tool Integration

1. Introduction

Dependable software is one of the most important factors in ensuring the safe operation of Cyber-Physical Systems (CPS) in an increasingly technology-driven society. To ensure that embedded software behaves acceptably, a number of *process* standards relevant to safety have been published, e.g. IEC61508:2010

*Corresponding author

Email address: fasplund@kth.se (Fredrik Asplund)

¹Phone: +46 8 790 7405, Fax: +46 8 20 22 87

(International Electrotechnical Commission, 2010), ISO26262 (International Organization for Standardization, 2011) and DO-178C (Special Committee 205 of RTCA, Inc., 2011). These emphasize all or part of the engineering activities that produce safety-critical systems of a certain type, rather than only the product features of the systems themselves.

Tool usage is a common part of modern engineering activities and may have relevant safety-related implications. Guidance with regard to tools and tool integration is therefore often provided by these standards. However, both the safety-related implications of tools / tool integration and the standards are difficult to analyze, e.g. due to complexity, the number of stakeholders involved, the few sources of quantified evidence available, etc.

The question addressed in this paper is whether the *guidance* with regard to the *integration of software tools* provided by high profile *systemic safety standards*² for CPS in the transportation domain is sufficient, or incomplete.

Validity is discussed in detail in Subsection 3.2, but even here in the introduction it can be noted that the difficulties of studying standards are of necessity evident in the research design. Proving the success of a safety standard is difficult, since the many possible confounding factors make it hard to state that a causal relationship that holds in one context will also hold in the next. However, in cases where a causal relationship (between an end product and some part of the related engineering activities) is regarded as more or less unimportant by the relevant safety standard, there *is* value in identifying associated safety-related implications. This type of negative evidence namely points to an omission by the safety standard in question. The approach chosen in the study presented in this paper is therefore to investigate the causal relationships between a set of safety-related characteristics of tool chains (see Section 3.1) and *software faults* found during development (in-house faults) and after release (field faults). Faults are flaws or omissions that may lead to unintended or unanticipated effects in the operational system. When the guidance in the surveyed systemic safety standards is not designed to identify some of the observed faults influenced or caused by tool usage, then clearly further discussion is required on the type and severity of this omission.

The related *State of the Art* is discussed in Section 2. Section 3 then considers the background, approach and validity of the study presented in this paper. The findings of the study are presented in Section 4 and discussed in Section 5. The paper closes by summarizing the conclusions in Section 6.

2. State of the Art

This section presents the academic and industrial discussions related to the question addressed in this paper.

²With systemic safety standards the author refers to standards that either a) require issues relevant to the entire life-cycle of the system under development to be taken into account, or b) are commonly deployed together with standards that ensure this.

2.1. An Academic Perspective

Several safety-critical domains, such as automotive, avionics and railways, have been frequently used as settings for research into tool integration. However, it is not the safety of the tool integration *per se* that has been studied.

Rather the primary focus is the importance of the *development life-cycle functionality* that a highly integrated support environment can supply. Examples include the RACME framework (Ceccarelli et al., 2011) and tailored tool-chain instances of the CESAR RTP (Armengaud et al., 2011). The *services* of a group of carefully selected and integrated tools is also a common focus - for example, to emphasise how the combination of diverse tools can support an otherwise unlikely range of design and verification aspects (Gönczy et al., 2009).

The discussion regarding reference workflows by Conrad et al. (2010) is one of the research efforts on tool integration that comes closest to dealing with the safety-related implications of the tool integration itself. They proposed the use of pre-qualified tools and associated workflows to limit a required qualification, thereby encompassing the entire tool-chain when considering safety.

With such a weak focus on the safety-related implications of tool integration within the research field, one can look at other fields in an effort to identify similar problems and viable approaches. Development organizations make up their own type of system, where the properties defining what is safety oriented are frequently different to those with importance in an operational system. Additionally, when development and operational use is discussed together it is often with regard to how the former influences the latter. Trying to transfer findings from operational systems to the context of development can therefore be precarious, as shown by for instance by Rollenhagen (2010) with regard to safety culture. Nevertheless, if one is careful with what is inferred, the transfer of explanatory models can at least be explored. To this effect models from the *automation* research field are of interest, since software tools essentially provide the automation of human activity (in the context of this paper with regard to engineering activities).

The early, much cited report on automation by Fitts et al. (1951) may be more balanced than it is given credit for, but it does reason in terms of what humans and machines can do better than each other. Later research has instead focused on how automation commonly alters human activity rather than making it obsolete (Parasuraman et al., 2000) and that humans and machines are not interchangeable but complementary (Billings, 1997). This broadens the common but narrow view that automation only refers to some kind of system that is acting independently. Sheridan's often used ten degree scale starts with no automation and ends with fully independent automation (Sheridan, 1992), but between these lowest and highest levels are steps at which the automation stops short of this - instead e.g. offering suggestions and informing the operator of actions taken. Parasuraman et al. (2000) details the understanding of automation further by applying Sheridan's scale to different types of automation in a model of human-automation interaction. The different types include *acquisition automation* (automation related to the sensing and registration of input

data, such as highlighting and filtering), *analysis automation* (automation of working memory and inferential processes, such as predictions and integration of data), *decision automation* (automation related to a selection from among decision alternatives) and *action automation* (automation related to the execution of the action choice). It is worth noting that a single system may exhibit (safety-related) flaws due to *too much* or *too little* automation related to any or several of these types.

2.2. Industrial Best Practices

Hundreds of safety standards exist, making it infeasible to survey them all. The *transportation domain*, which as previously mentioned has been frequently used for studying tool integration, is however a suitable limitation. The safety-related implications of best practices concerning *separate* tools in this domain have been given considerable attention (see e.g. the summary primarily focused on aviation by Kornecki and Zalewski (2009)); the transportation domain is also likely to influence other CPS domains in the future since it is deemed highly important by the European Union (ARTEMIS ITEA Cooperation Committee, 2013).

The transportation domain has high profile systemic safety standards such as DO-178 (DO-178B and DO-178C), IEC 61508 (IEC 61508:1998 and IEC 61508:2010), BS EN 50128 (BS EN 50128:2001 and BS EN 50128:2011) and ISO 26262. Several of the standards differ, e.g. due to differences in their basic assumptions and the stakeholders involved during their creation; DO-178 is a software assurance standard, while ISO 26262 deals with the entire safety lifecycle of automotive safety-related systems comprised of electrical, electronic and software components; ISO 26262 is considered a goal-based standard, while IEC 61508 is considered prescriptive. Upon closer examination, however, several commonalities can be identified with regard to both separate tools and tool integration.

For a full review of these commonalities (and differences) the reader is referred to Asplund (2014). To summarize one can identify two views on how to ensure trust in tools and tool integration. The first approach is found in e.g. DO-178C. Trust in an end product is engendered by the constraints on its development process. In the same way, trust in a tool used during safety-critical development is ensured by its development process constraints. Tool integration is treated in a bottom-up manner, since it is considered per tool with regard to how strict tool qualification is required (assumptions about tool integration may allow or disallow tool qualification at different qualification levels). The second approach is found in e.g. IEC 61508:2010. Here trust is established by generic measures within the engineering environment of the targeted CPS, such as securing thorough specifications and performing tool assessments. The focus is mostly on the reliability of a specific set of tools, while tool integration is even referred to in a simplistic manner as something that purely minimizes the probability of operator error. These approaches do not preclude a top-down approach on tools and tool integration. High level guidelines in both DO-178C and IEC 61508:2010 in fact underline considerations such as the consistency

and complementarity of tools throughout the development life-cycle. However, neither approach provides explicit guidelines on how to ensure trust throughout a whole tool chain.

These standards go beyond a simplistic discussion of technology, employing a perspective that can be summarized as viewing automation through software tools as a “team member” (Pritchett (2009) provides a more detailed discussion of this type of perspective). As long as the rest of the standards are adhered to the view is that this type of automation can and should be evaluated based on its reliable execution of separate process steps independent of human operators. Software tool automation that only supports the actions and decision-making of operators is seen as relatively inconsequential. With regard to the different types of automation described in the previous subsection these standards therefore seem to put the emphasis on software tools that provide high levels of action automation. Even if highly skilled and professional operators might catch problems related to low levels of acquisition and analysis automation, it seems *improbable* that these safety standards will provide much support in this regard.

3. Research Design

This section starts by presenting the background and explaining the approach of the study, then closes with a discussion of the validity and limitations of the research findings.

3.1. Background

iFEST (iFEST Consortium, 2013) was an ARTEMIS Industrial Association research project focusing on the specification and implementation of an integration framework to establish and maintain tool chains for the engineering of complex industrial embedded systems. An exploratory case study focusing on the support environment of an iFEST industrial partner led to the proposition that there are certain safety-related characteristics of tool chains. The reader is referred to Asplund et al. (2012) for a description of the individual characteristics and how they were identified. To summarize, they included *Traceability for Completeness and Consistency* (TfCaC), *Well Defined Data Semantics* (WDDS), *Customized GUI's* (CGUI), *Coherent Time Stamp Information* (CTSI), *Process Control/Notifications* (PCN), *Automated Transformations of Data* AToD, *Automated Tool Usage* (ATU), *Data Integrity* (DI) and *Data Mining* (DM).

No particular order of priority was attributed to the characteristics at this point in time, beyond that weaknesses in support environments related to any one of them could lead to the development of unsafe end products. However, safety standards can potentially restrict the use of tools and tool chains. A particular characteristic could therefore be extensively discussed and important in theory, but irrelevant in practice (or vice versa). The significance of the characteristics then lies in the way they summarize risk so that it can more

readily be identified. Using the list of characteristics, even though it may not be complete, allows a broad search for the omissions hinted at in Subsection 2.2 in line with the approach described in the beginning of the paper.

This effort to identify omissions in standards could therefore take part in one of three tracks (separate, smaller studies) in a *sequential mixed model study* in two phases. Tashakkori and Teddlie (1998) provides a description of this type of study. Two large international companies were studied within each of the three tracks. We received access to their fault data and employees, providing that liability issues were considered by not further disclosing the company name or the magnitude of the fault numbers.

3.1.1. The Two Different Phases

The first, quantitative phase consisted of the analysis of about 1200 software defect reports from the defect databases at the two companies. The software faults detailed by the defect reports were quantified based on the characteristics defined in the previous subsection, in order to identify:

- Indicators that the characteristics had an influence, even when no faults could be directly linked to them. The related findings are presented in Subsection 4.1.
- Weaknesses of the support environments related to the characteristics that could be proven to have caused faults, especially safety-critical faults in software released to customers. The related findings are presented in Subsection 4.2.

Based on the results from the analysis, a second phase was conducted to follow up on *alternative explanations*. This phase consisted of interviews with employees that had experience from a number of roles at each company, resulting in 7 employees at Company A and 6 employees at Company B being interviewed. Questions were asked to provide information on undocumented faults related to tool integration, faults related to tool integration found in other projects than those sampled, and the context of the relevant faults already identified.

3.1.2. The Companies

Company A develops CPS in which the software has direct implications on safety, i.e. in which software failures may lead *directly* to physical harm. The company was chosen because it is large, continuously evaluated against systemic safety standards and has an excellent safety record. Company B is also a large, worldwide company. This company develops end products in which the software has no direct implications on safety, since mechanical safeguards protect the operator from any hazardous effects of software failures. The product studied at Company B has, like the product studied at Company A, been active for many years and is of no trivial complexity and size. Otherwise the companies operate in very different application domains even though they need to fulfill similar high level requirements, such as adhering to strict timing demands, minimizing

downtime and handling physical substances with exactness. In the track (smaller study) presented in this paper Company B was of minor importance, since the focus was on tool usage during safety-critical development. This company was therefore mainly used to support the search for alternative explanations. However, for the sake of completeness, transparency and the requirement not to divulge the magnitude of any fault numbers, the related research data is anyway presented in its entirety in the subsequent parts of the paper.

3.2. Ensuring Validity

The subsequent three subsections discuss the validity and limitations of the different phases of the study.

3.2.1. First Phase

A number of issues had to be considered prior to the first phase of the study to ensure the internal and external validity and the reliability of the results.

The majority of the defect reports were from two major projects, one at each company. The accessible population of defect reports from Company A's project was limited, since it had only just passed the milestone indicating a complete design specification. This project, as well as the corresponding project at Company B, was chosen *despite this*, since it allowed easy access to experts directly involved in the development (thereby limiting observer bias). To limit the possibility of sampling bias, only defect reports prior to an equivalent milestone of Company B's project were included in the rest of the study. To limit the possibility of selection bias two additional analyses were therefore performed on previously completed projects at each company. The first involved minor projects, which conducted limited updates to the functionality of a pre-existing product. The second involved major projects and the complete set of defect reports concerning software released to customers.

Faults are directly observable and, in those cases when weaknesses of support environments are related to the characteristics and have been found to cause faults, the internal validity is primarily established through precautions with regard to observer bias. The same is true for mislabeled defect reports dealing with other considerations, such as feature requests, which could easily be excluded from the study. With regard to the indicators, the question is more complex due to the choice to minimize observer bias, since it limited the accessible fault population to only those from a part of the major projects. Here the reasoning is therefore further supported through the extended analysis of historical projects.

Two problems related to the external validity is especially pertinent in this study. Firstly, considering that we are looking for evidence of a casual relationship that is not considered by safety standards, are the involved companies special in any way that might lead to such a causal relationship even though it is otherwise improbable? The choice of the companies is decisive with regard to this problem. Neither of the companies employ technology that cannot be expected to be found at other companies of similar resources, with the State

of the Art being far ahead. If any of the characteristics exert an influence at these companies, they can be expected to be able to do so at other best practices companies. However, the study focuses on software faults within CPS. The findings are therefore not automatically transferable to engineering disciplines other than software engineering or to systems other than CPS. Secondly, the safety standards surveyed in Subsection 2.2 are all within the transportation domain. Even though these standards spend much effort on the subject of tool qualification, there might be other domains with safety standards that lack the identified omissions. The discussion and conclusions have been suitably limited to take both of these problems into account.

The threat to reliability in this phase is also likely to be related to observer bias, i.e. that other observers would evaluate the data differently. To avoid this bias, different employees at the two companies were asked to review and provide their views on all ambiguous observations. This essentially happened every week of the study, especially in regard to judging the criticality and effect of a fault.

3.2.2. Second Phase

The second phase was also planned to ensure the validity of the results, but the choice of the interview method required different considerations (Kvale and Brinkmann, 2009). Firstly, a field journal was used e.g. to document questions. Secondly, the choice of interviewees was based on the intent to cover the perspectives of as large a part as possible of the development organization. The developer, project manager, manager, tester, designer and support environment customizer roles were covered at both companies. Thirdly, the ethical considerations discussed prior to the study included potential reasons why interviewees might provide biased or incorrect information. To avoid this, the interview transcripts were approved by the interviewees after completion and a more confrontational interviewing technique adopted. Fourthly, to avoid the risk of interviewer bias, secondary interviewers were present during early interviews.

3.2.3. Interphase Considerations

The choice of research design was mainly based on the intent to increase the quality of inferences. According to e.g. Tashakkori and Teddlie (1998) the possibility that alternative explanations exist is decreased by use of a triangulation of quantitative and qualitative methods.

4. Research Findings

The research findings of the study are presented in two subsections, respectively addressing the two categorisations described in Subsection 3.1.1. As mentioned in Section 3, concerns regarding legal liability prohibits the magnitude of fault data to be divulged. The research data is therefore provided in percentages. The discussion found in the subsequent sections takes this into account.

Figure 1: Company A, Indicators in Percentages

Figure 2: Company B, Indicators in Percentages

4.1. Indicators of the Characteristics

13% of the faults from the early part of the main project at Company A showed indication of being influenced by weaknesses in the support environments related to the characteristics. For field faults the percentage was 1%, while for the late iteration the percentage was 14%. The percentages from Company B were 3%, 2% and 16% respectively. For those characteristics that were found to exert an influence, the distribution of indicators (further divided into positive and negative) are shown in Figures 1 and 2. Negative influence has in these cases been identified by finding a weakness in a support environment associated with a characteristic in a defect report part describing the cause or handling of a fault. Positive influence has been identified by finding a characteristic mentioned in a defect report part describing how a fault was found.

The distributions from the three different populations at each company show little similarity. Due to the low number of indicators in late iterations and software released to customers, it is only possible to perform a chi square goodness of fit test when comparing the main project and the late iteration from Company A³. This test, however, allows one to conclude that the influence of the characteristics varies across the development life-cycle, since the early parts of the main project cannot be used to predict the distribution of indicators in the

³The problems in performing statistical tests were not unexpected. If one considers the previously completed project at Company A from where faults were sampled, the field faults make up only 1% of the total faults and the faults from the large, late iteration less than 5%.

Figure 3: Company A, Causes in Percentages

late iteration.

The interviews uncovered references to faults in other projects which also indicated an influence by the characteristics. Additionally, when questioned about problems related to tools and tool integration, the interviewees could often provide examples. Almost all interviewees had experienced problems when trying to understand GUI's (all 7 employees at Company A and 4 out of 6 at Company B). Most of the interviewees had experienced problems with extracting or viewing data from the development process (6 and 3 interviewees, respectively). A substantial part of the interviewees had also encountered reliability issues (4 and 1 interviewees, respectively) or the incorrect handling of notifications (3 and 1 interviewees, respectively). However, none of the interviewees could provide an example of when their encounters with such problems had resulted in a fault in the end product (although some gave examples of when this had happened to other employees). At most, these problems had resulted in an extra work effort by the interviewees to correct temporarily erroneous artifacts. Most of these problems were also related to tools, with only a few examples involving tool integration. In fact, while all interviewees considered tools an essential part of their work environment, only half of them considered any tool integration important. The rest of the interviewees considered tool integration either unimportant or something that provided some helpful but not critical support.

4.2. Faults Caused by Weaknesses in Support Environments

The number of faults of which weaknesses in a support environment related to the characteristics are described as the cause make up 1% of the surveyed faults (versus 11% showing an indication of the characteristics). Their distribution is shown in Figures 3 and 4.

All of these characteristics were related to several weaknesses that had caused faults:

- *Traceability for Completeness and Consistency.* Firstly, when there was a lack of links between requirements belonging to different abstraction levels, the resulting confusion could lead to mistakes. Secondly, when temporary links to “dummy” requirements were not updated, completeness could be falsely assumed.

Figure 4: Company B, Causes in Percentages

- *Automated Tool Usage.* Firstly, when developers relied on automatic testing to check whether software was reliable enough to be released, these automatic tests could fail to detect flaws. Secondly, when distributed building of code resulted in different build servers labeling binaries differently, this could result in confusion among operators.
- *Process Control/Notifications.* Firstly, when some artifacts were changed without relevant stakeholders being notified, the result could be asynchronous evolution of development artifacts. Secondly, when weaknesses in the code were identified during reviews, the complete set of weaknesses was sometimes not acted on.

The interviews uncovered references to faults in other projects with *similar* causes, but also:

- *Customized GUI's.* When comparing the differences between different versions of code prior to a release, the sheer number of changes could make their impact difficult to assess. One mentioned result was the erroneous update of a system configuration that led to the removal of a critical part of a user interface.
- *Data Integrity.* When distributed building of code erroneously used different compiler versions, this sometimes led to incompatible binaries.

Of the characteristics-related faults, only those related to the *Customized GUI's* characteristic and the first type of the *Process Control/Notifications* characteristic were safety-critical. The former was also a field fault.

5. Discussion

The following three subsections discuss and elaborate on the research findings presented in the previous section. The first subsection summarizes the most important observations of the study and how these relate to the two categorisations described in Subsection 3.1.1. The second subsection, based on the literature on support environments for CPS, presents the likely future development

of support environments in areas related to the most important observations. The third subsection builds upon the first two to discuss whether modern safety standards contain omissions that will have an increasing possibility of negatively influencing the safety of end products if support environments become more and more automated.

5.1. What is observed?

An important observation is that weaknesses in support environments related to the characteristics could lead to software faults through *a too high level of action automation*, in the form of erroneous or incomplete *Automated Tool Usage*. Automation has independently caused and failed to identify faults, which gives credence to the “team member” perspective on automation put forth by the safety standards surveyed in Subsection 2.2.

A development organization itself also makes this perspective natural. All interviewees acknowledged that they could not perform their work without the support of modern software tools. At the same time most of them had experienced frequent problems with the reliability of and interaction with these tools, but without these problems leading to any apparent adverse effects (save some extra effort “to make things right”). Due to the essential nature of the tools, the interviewees had tried to adapt to make up for any weaknesses in them. The only unacceptable weakness would thus be tools introducing faults in ways that interviewees could not compensate for - in other words, if tools introduced faults through automation outside the control of the operator. This acceptance of weaknesses may also be the reason why tool integration was deemed unimportant by so many interviewees. As long as data was shuffled around between tools in a reliable way, the interviewees thought they could deal with any shortcomings of the tool integration on their own.

However, in the cases when tool integration influenced the end product negatively during safety-critical development, it was more often in an indirect rather than a direct fashion. Firstly, traceability was identified both as the primary positive and the primary negative influence at Company A. Secondly, the only identified safety-critical *field* fault caused by a weakness in a support environment related to a characteristic was introduced through a GUI not being suitable for a particular task during development. One may also note that, regardless of the strong focus on information management stipulated by system engineering, safety-critical faults were also introduced through the lack of well designed notifications and process control.

Another important observation is thus that weaknesses related to the characteristics could lead to software faults through *a too low level of acquisition and analysis automation*, which this study shows for lack of appropriate *Customized GUI's* and missing *Process Control/Notifications*, and makes likely for manually implemented *Traceability for Completeness and Consistency*.

5.2. What can be expected?

It is then natural to ask oneself how likely this is to be important in the future of safety-critical CPS development.

With regard to the implications of a too high level of action automation, more fully independent automation in safety-critical development is to be expected in the future. Automated verification in particular has been increasingly used in safety-critical development (Broy et al., 2010). However, systemic safety standards seem to identify and address this potential problem.

Automation of tools and tool integration at the low end of Sheridan’s scale, which supports the human operator instead of replacing him, is also likely to increase. As an example, both simulation (Benveniste et al., 2005; Kang et al., 2013) and analysis (Broy et al., 2010; Lahtinen et al., 2012) are already used extensively in safety-critical development. The associated tools and tool integration will most likely evolve further during the next decade together with support to mitigate many other important issues at a low level of automation (such as filtering of changes and notifications to avoid information overflow (Törnngren et al., 2008)). By failing to address these potential problems, some systemic safety standards allow for overconfidence in low levels of automation.

5.3. What needs to be done?

Tool integration aimed at supporting operators through acquisition and analysis automation will become critical to ensure that engineers keep making the right choices with regard to an increasing number of complex system-wide properties. This is not a question of replacing process steps in future safety-critical development, but rather one of supporting the reasoning of the individual operators. Building from a bottom-up, “team member” perspective, the surveyed systemic safety standards have little in the way of guidelines for establishing trust at these low levels of automation.

When even obviously flawed tools and tool integration can be accepted as long as the operator thinks he can ordinarily compensate for these flaws, and with low levels of automation leading to safety-critical field faults, there is a need for further guidance by the surveyed systemic safety standards. Other safety standards that do not already provide this guidance or regulate a safety-critical domain in which these concerns can be proven to be minimal are also in need of change. This paper proposes the introduction of *software tool chain qualification* to ensure that erroneously designed (or incomplete) acquisition and analysis automation does not lead operators astray.

With operators acting on the complete tool chain to establish an understanding of the current state of development, such software tool chain qualification must be focused at a higher level of organization than that of individual software tools. This calls for each *task* in which software tools are used within development to be evaluated according to the following criteria:

- The possibility that the combination of all software support for a task is unsuitable for bringing faults or omissions in the safety-critical system being developed to the attention of operators during development.
- The confidence in preventing or detecting such faults or omissions.

Considering that standards are a compilation of best practices, the introduction of such tool chain qualification guidelines is probably still some years ahead. In the meantime this omission may act as a guide for safety-related research within the tool integration research field. One should in that case keep in mind that tool integration is a research field that could benefit from theory triangulation to foster the use of a more diverse set of research methods (Asplund, 2014).

Research fields closely related to tool integration offer some possibilities in this regard, since the identified problems could be related to such things as human-machine interaction and process management. However, a more distant research field could potentially be more fruitful with regard to theory triangulation. Given that the identified omission concerns low levels of automation, it is the author's suggestion that *organizational research* should be approached, especially in relation to *safety culture*. The term figures in industry and there often implies - as exemplified in the description given in ISO 26262 - that processes for continuous improvement need to be in place. If the safety-related implications of tools and tool integration increase as support environments become more automated, these processes should have direct bearing on the outcome with regard to risk. Due to the distance between the research fields there are, however, difficulties that must be overcome to make this a reality. The theoretical implications of the term safety culture for instance is diverse, since it is a term with many different definitions in many different contexts, as noted by e.g. Guldenmund (2000). Edwards et al. (2013) describe three types of conceptualisations of the term: normative, anthropological and pragmatist. Which of these is most easy (or even possible) to adopt during theory triangulation with tool integration is an open question. Furthermore, improper tool integration may lead to risks, but the associated faults may only rarely lead to accidents and may not always be the triggering factor even when they could be. Proving causal relationships may be difficult when it is not straight-forward as to which type of rule-breaking, attitudes or behaviour leads to risk.

6. Conclusions

The study presented in this paper, focusing on the best practices in CPS software development, identifies a negative, indirect influence from - as well as faults caused by - both too high and too low levels of automation in support environments.

The transportation domain is a high profile CPS domain that has frequently been used for studying tool integration and the safety-related implications of best practices concerning separate tools. Nevertheless, modern high profile safety standards in this domain advocate essentially bottom-up approaches for tool qualification. These approaches do not handle tool usage problems at low levels of automation when operators depend on tool integration to give critical support in assessing development processes or artifacts.

The leaps in technology expected to occur in the near future, due to demands for increased efficiency and cost reduction, are likely to lead to an increased

dependency on tools and tool integration providing a low level automation. The next generation of standards for safety-critical development of CPS that do not provide guidance beyond the high profile safety standards of the transportation domain therefore need to change. They should include guidelines for a top-down software tool chain qualification process that starts with tasks rather than technology. Otherwise the introduction of modern support environments runs the risk of pushing operators towards erroneous or incomplete decision making.

Given the type of omission it is suggested that researchers focusing on the safety-related implications of tool integration approach organizational research in search of possibilities to set up theory triangulation studies.

Acknowledgments

Special thanks go to Vicki Derbyshire for her help in proofreading this paper, and to Jad El-khoury and Martin Törngren for their advice on how to improve certain parts of it.

References

- Armengaud, E., Zoier, M., Baumgart, A., Biehl, M., Chen, D., Griessnig, G., Hein, C., Ritter, T., Kolagari, R.T., 2011. Model-based toolchain for the efficient development of safety-relevant automotive embedded systems, in: Proceedings of the SAE 2011 World Congress & Exhibition. URL: <http://papers.sae.org/2011-01-0056>.
- ARTEMIS ITEA Cooperation Committee, 2013. ITEA ARTEMIS-IA High-Level Vision 2030, Opportunities for Europe. Technical Report. ARTEMIS Industry Association & ITEA Office Association.
- Asplund, F., 2014. Risks Related to the Use of Software Tools when Developing Cyber-Physical Systems. Ph.D. thesis. KTH Royal Institute of Technology.
- Asplund, F., El-khoury, J., Törngren, M., 2012. Qualifying software tools, a systems approach, in: Ortmeier, F., Daniel, P. (Eds.), Computer Safety, Reliability, and Security, 31st International Conference, SAFECOMP 2012, pp. 340–351.
- Benveniste, A., Jonsson, B., Buttazzo, G., Thiele, L., 2005. Embedded Systems Design, The ARTIST Roadmap for Research and Development. volume 3436 of *Lecture Notes in Computer Science*. Springer-Verlag. doi:10.1007/b106761.
- Billings, C.E., 1997. Aviation Automation, The Search for a Human-Centered Approach. Lawrence Erlbaum Associates, Inc.

- Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S., Ratiu, D., 2010. Seamless model-based development: From isolated tools to integrated model engineering environments. *Proceedings of the IEEE, Special Issue on Aerospace and Automotive Software* 98, 526–545. doi:10.1109/JPROC.2009.2037771.
- Ceccarelli, A., Vinerbi, L., Falai, L., Bondavalli, A., 2011. RACME: A framework to support V&V and certification, in: 2011 5th Latin-American Symposium on Dependable Computing (LADC), pp. 116–125. doi:10.1109/LADC.2011.21.
- Conrad, M., Munier, P., Rauch, F., 2010. Qualifying software tools according to ISO 26262, in: Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VI, pp. 117–128.
- iFEST Consortium, 2013. industrial framework for embedded systems tools. URL: <http://www.artemis-ifest.eu/>.
- Edwards, J.R., Davey, J., Armstrong, K., 2013. Returning to the roots of culture: A review and re-conceptualisation of safety culture. *Safety Science* 55, 70–80. URL: <http://www.sciencedirect.com/science/article/pii/S0925753513000088>, doi:<http://dx.doi.org/10.1016/j.ssci.2013.01.004>.
- Fitts, P.M., Viteles, M.S., Barr, N.L., Brimhall, D.R., Finch, G., Gardner, E., Grether, W.F., Kellum, W.E., Stevens, S.S., 1951. Human Engineering for an Effective Air Navigation and Traffic Control System. Technical Report ADB815893. Ohio State University Research Foundation.
- Gönczy, L., Majzik, I., Horváth, A., Varró, D., Balogh, A., Micskei, Z., Pataricza, A., 2009. Tool support for engineering certifiable software. *Electronic Notes in Theoretical Computer Science* 238, 79–85.
- Guldenmund, F., 2000. The nature of safety culture: A review of theory and research. *Safety Science* 34, 215 – 257. URL: <http://www.sciencedirect.com/science/article/pii/S092575350000014X>, doi:[http://dx.doi.org/10.1016/S0925-7535\(00\)00014-X](http://dx.doi.org/10.1016/S0925-7535(00)00014-X).
- International Electrotechnical Commission, 2010. BS/IEC 61508:2010, functional safety of electrical/electronic/programmable electronic safety-related systems.
- International Organization for Standardization, 2011. ISO 26262:2011, road vehicles - functional safety.
- Kang, E.Y., Enoiu, E.P., Marinescu, R., Seculeanu, C., Schobbens, P.Y., Pettersson, P., 2013. A methodology for formal analysis and verification of EAST-ADL models. *Reliability Engineering & System Safety* 120, 127–138. URL: <http://www.sciencedirect.com/science/article/>

- pii/S0951832013001725, doi:<http://dx.doi.org/10.1016/j.res.2013.06.007>.
- Kornecki, A., Zalewski, J., 2009. Certification of software for real-time safety-critical systems: State of the art. *Innovations in Systems and Software Engineering* 5, 149–161.
- Kvale, S., Brinkmann, S., 2009. *InterViews, Learning the Craft of Qualitative Research Interviewing*. 2 ed., SAGE Publications, Inc.
- Lahtinen, J., Valkonen, J., Björkman, K., Frits, J., Niemelä, I., Heljanko, K., 2012. Model checking of safety-critical software in the nuclear engineering domain. *Reliability Engineering & System Safety* 105, 104 – 113. URL: <http://www.sciencedirect.com/science/article/pii/S0951832012000555>, doi:<http://dx.doi.org/10.1016/j.res.2012.03.021>.
- Parasuraman, R., Sheridan, T., Wickens, C.D., 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 30, 286–297. doi:10.1109/3468.844354.
- Pritchett, A.R., 2009. Aviation automation: General perspectives and specific guidance for the design of modes and alerts. *Reviews of Human Factors and Ergonomics* 5, 82–113. URL: <http://rev.sagepub.com/content/5/1/82.full.pdf+html>, doi:10.1518/155723409X448026.
- Rollenhagen, C., 2010. Can focus on safety culture become an excuse for not rethinking design of technology? *Safety Science* 48, 268 – 278. URL: <http://www.sciencedirect.com/science/article/pii/S095183201000143X>, doi:<http://dx.doi.org/10.1016/j.ssci.2009.07.008>.
- Sheridan, T.B., 1992. *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press.
- Special Committee 205 of RTCA, Inc., 2011. DO-178C, software considerations in airborne systems and equipment certification.
- Tashakkori, A., Teddlie, C., 1998. *Mixed Methodology, Combining Qualitative and Quantitative Approaches*. Sage Publications.
- Törngren, M., Chen, D., Malvius, D., Axelsson, J., 2008. *Automotive Embedded Systems Handbook*. CRC Press, Taylor & Francis Group. chapter 10.