Safety-Guided Design through System-Theoretic Process Analysis, Benefits and Difficulties

M.S. Fredrik Asplund, Ph.D. Jad El-khoury, Professor Martin Törngren; KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Stockholm, Sweden

Keywords: Tool Integration, Safety, STAMP, STPA, Safety-Guided Design

Abstract

Development environments for embedded systems are moving towards increased automation between *Commercial Of The Shelf* (COTS) engineering tools. While automation provides new opportunities for e.g. verification, it also to some extent decreases the possibility of identifying and acting on safety issues that arise during development. To investigate the relationship between tool integration and safety we performed a *System-Theoretic Process Analysis* (STPA) of a tool chain from an industrial case study. This tool chain was then reanalyzed and redesigned twice, in part motivated by identified hazards.

This paper presents our experiences from applying STPA to safety-guided design in the context of integrating COTS engineering tools into tool chains. We discuss the benefits of and difficulties with applying STPA. We also suggest improvements that complement STPA with support methods and tools.

The primary benefit was the support in categorizing risks and causes. The three difficulties we encountered were identifying context-specific causal factors, defining control structures across several domains (management, user, technical, etc.) and limiting the domains taken into account. The use of STPA during safety-guided design would be facilitated by the use of expert systems and simulation, especially in regard to relating different domains.

Introduction

The diverse expertise required to build tools for use when developing embedded systems requires development organizations to rely on *Commercial Of The Shelf* (COTS) engineering tools. COTS tools will require *tool integration* to enable the deployment of one or more sequences of tools (i.e. *tool chains*) to support the *development process(es)* at hand. Tool integration can be performed manually by *operators* (managers, developers, tool chain developers, etc.) or be supported by *tool integration mechanisms* (i.e. automation scripts, data transformation engines, traceability tools, etc.). The presence of at least one of these types of tool integration is necessary and both types of tool integration are associated with different benefits. It is for instance common for manual tool integration activities to bring together experts from different engineering disciplines to ensure optimal solutions. Similarly, automated tool integration can facilitate and improve the comparison of requirements with verification and validation activities (for completeness), decision support based on leading indicators for risks, etc. However, the presence of tool integration is not unproblematic, since the operators and software involved may handle development artifacts in an unsafe way or even introduce errors in them. This may lead to the introduction of unsafe states in the end product.

Many *prescriptive process standards* (ref. 11) deal with issues related to tool integration, since stipulating how to produce a product requires one to deal with issues like how data is transferred between development phases (ref. 6), how software tools that replace process steps need to be qualified (ref. 15), etc. We initially studied the relationship between tool integration and safety based on the primarily prescriptive process standard ISO 26262 (ref. 3), but found our results overly restricted. While ISO 26262 highlights the importance of many parts of tool integration in regard to safety, it does so from the viewpoint of the automotive domain and with the objective to ensure *sufficient*[1] safety. We could not be sure of the generality or the completeness of our results. We therefore decided to look at *Hazard Analysis Techniques* (HATs) motivated by the way HATs can be used to study accidents *before* they occur. By analyzing a tool chain from an industrial case study using an appropriate HAT we expected to find risks that link tool integration to safety. These risks could then form the basis for discussing a more generic and complete perspective of the relationship between safety and tool integration.

[1] A system can usually never be completely safe, since it is impossible to control all phenomena in the physical world. Safety standards are therefore essentially a compromise between ensuring safety *sufficiently* well and for instance the implied cost of development, operations, maintenance, etc.

This analysis was carried out and yielded useful results, which are described together with the details of the case study in reference 1. The objective of this paper is to discuss and expand on our experiences with the *System-Theoretic Process Analysis* (STPA) HAT to perform safety-guided design. In the next section we discuss the reasoning behind our choice of HAT. This is followed by a section that describes the basics of STPA (i.e. our choice) and a section that describes our customization of STPA to yield relevant results within the context of tool integration. The three subsequent sections discuss the benefits of applying STPA, the problems related to applying STPA and suggestions for improvements to the support for STPA that we identified. After this we close with a section summarizing our conclusions.

A Suitable Hazard Analysis Technique

The nature of an embedded systems development effort places strict requirements on the properties of a HAT used to analyze it. Any large development effort is complex, since it involves several different experts interacting across different development phases (project managers, requirements engineers, testers, etc.). If there are safety considerations this adds to the complexity by introducing restrictions not only on the development activities, but also on the whole organization (i.e. guidelines for the establishment of a safety culture, time plans, etc.). Embedded systems development has the extra problem that it involves experts from many different engineering disciplines (for instance aerospace engineers, electric engineers, mechanical engineers, etc.). This makes the use of *Failure Modes and Effects Analysis* (FMEA) and *Fault Tree Analysis* (FTA) (ref. 16) problematic. While these HATs are widely used, they both focus firmly on distinct failures and how these propagate through a system. This approach is of limited help when the "root causes" identified are merely symptoms of underlying problems, which is often the case in systems as complex and abstract as an embedded systems development effort[2]. *Hazard and Operability Study* (HAZOP) (ref. 16) is slightly better suited with its approach of using a series of guide words for constructing hypothetical, safety-related questions about systems. However, HAZOP relies on iterating through the different parts of a system and their relationships one by one using a pre-defined structure of the system. The dependencies between the different parts of an embedded systems development effort can be implicit, consist of several different parts acting in unison and involve entities of very different domains (from organizational roles to software tools). HAZOP could be combined with other HATs to better handle some of these issues (*Layer of Protection Analysis* (LOPA) (ref. 4), for instance, would require that all domains of importance are at least defined), but the approach of combining several HATs would require a large expert effort in itself.

Due to three key benefits our choice instead fell on the *System-Theoretic Process Analysis* (STPA) HAT. The first benefit is that STPA is based on the *Systems-Theoretic Accident Model and Processes* (STAMP) causality model (ref. 10) and therefore takes new causal factors, identified in STAMP, into account. These factors "include design errors, including software flaws; component interaction accidents; cognitively complex human decision-making errors; and social, organizational, and management factors contributing to accidents" (ref. 10). This benefit was especially important in our context, due to the complex organizational and management factors mentioned in the previous paragraph. The second benefit is that STPA provides guidance throughout the analysis akin to the HAZOP HAT, but by the use of logical block diagrams rather than plans of constructs in the physical world. This makes the problems mentioned in relation to HAZOP in the previous paragraph less of an issue. The third benefit is that STPA does not rely on any part of the system at hand being pre-defined before the analysis can start, therefore allowing STPA to be used for safety-guided design. This was important for the set up of our case study and will be discussed in the *Customizing STPA to Analyze the Tool Integration of an Industrial Tool Chain* section.

A Short Introduction to System-Theoretic Process Analysis

The underlying idea behind STAMP is that the assumptions of many *State of the Practice* approaches (like those made in the implicit causality model of FMEA, FTA, etc.) to ensuring safety are no longer (if they ever were) valid. Instead of building analyses on the assumption that accidents are caused by chains of directly related events and that these events occur simultaneously by chance, STAMP highlights the fact that accidents occur because systems *allow* them to occur. Systems are modeled as interacting components that form closed control loops to enable a system in a state of dynamic equilibrium. Safety emerges when the appropriate constraints are enforced on the behavior of the

[2] As an example we can take an airbag that does not inflate fast enough in some extreme cases. The "root cause" for this could be found to be the use of the wrong version of a development artifact by a software engineer. However, this does nothing to help answer the important question of *why* the software engineer used the wrong version.

components, while accidents occur when these *system safety constraints* are allowed to be violated. An accident is therefore often preceded by a period of time during which the system transitions into a sequence of states in which an accident becomes more and more likely[3]. This leads to a different terminology then that of the commonly used *Fault-Error-Failure chain* (ref. 16). The most important difference is the definition of a *hazard*, since accidents are not seen as caused by chains of events a hazard is defined as *a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (a loss).* This also leads to *control loops* (i.e. the way the different components in the system can exert control and receive feedback to avoid hazardous states) and *risks* (i.e. the potential for transitioning into a hazardous state) becoming central to STPA, something which is reflected by the two main steps of STPA (the description below is based on reference 10).

Step 1 of STPA: The objective of step 1 of STPA is to identify the potential for inadequate control of the system that could lead to a hazardous state. An analyst therefore first has to choose which hazards to analyze[4]. Based on these, a *control structure* with all relevant system components and their interactions needs to be defined[5]. From the interactions in the control structure the *control actions* (i.e. those component interactions that exert control) are separated out and analyzed to see if any *basic*[6] or *additional*[7] *risks* could result in a hazardous state. If the context is suitable and the analysis limited the control actions can be analyzed using the basic (and any additional) risks directly. However, in more complex contexts this may not be an option, since how to apply these risks may then not be directly obvious. In this case the basic risks can be translated into *general types of risks*[8] based on the context being analyzed. Regardless of the complexity the objective is to eventually identify the *programmatic risks*[9] of the system at hand.

Step 2 of STPA: The objective of step 2 of STPA is to determine how each potentially hazardous control action identified in step 1 could occur. The first part of this activity is to identify the causes to the programmatic risks identified in step 2 by examining the control loops that the programmatic risks belong to. This is guided by STPA by use of general causal factors defined for the various parts of a generic control loop (Figure 1). The second part of step 2 is to consider how the designed controls could degrade over time and protect against this. After this one can return to step 1 again to update the control structure to eliminate the causes to the identified risks. In this way the safety can be increasingly assured for each of the iterations performed.

---

[3] The difference is most easily explained by use of a simple example. Imagine a cowboy who is performing gun practice by throwing his hat into the air and shooting at it. After shooting five bullets, his sixth bullet hits an innocent bystander. Analysis techniques such as FTA and FMEA will provide guidance towards the conclusion that shooting a gun is dangerous to this or that degree and leave the rest of the analysis to the expertise or common sense of the analyst. In STAMP the event of shooting the gun is not as important as the control exercised in the scenario in which it occurs. The first five shots may all be perfectly safe, but since nothing is stopping gravity from pulling on the hat the last shot will eventually be fired horizontally to the ground (which is unsafe). STAMP therefore provides guidance towards asking relevant questions in regard to the lack of control of the hat, but also in regard to for instance why there is no sheriff present to control gun practice in the proximity of innocent bystanders.

[4] Based on the previous example this might be the hazard that a cowboy performing gun practice is going to shoot an innocent bystander.

[5] Again, for the previous example this might include the cowboy, his gun, his hat and the bystanders, but it might just as well contain the sheriff, the courts or even the government (if deemed relevant).

[6] STPA identifies four *basic risks* that describe the possibilities for inadequate control, namely that (1) a control action required for safety is not provided or not followed, (2) an unsafe control action is provided, (3) a potentially safe control action is provided too early or too late and (4) a control action required for safety is stopped too soon or applied too long.

[7] STPA also allows for the definition and use of *additional risks* based on the nature of the context studied. An example of additional risks are the *coordination risks*, i.e. that (1) uncertainty regarding which controller should perform a control responsibility leads to a control action not being performed or (2) conflicting control actions from different controllers have unintended side effects.

[8] General types of risk are interpretations of the basic and additional risks based on the context and a specific hazard.

[9] Programmatic Risks are the specific unsafe control actions (in a specific context) in a control structure that can lead to the relevant basic, additional or general types of risks.

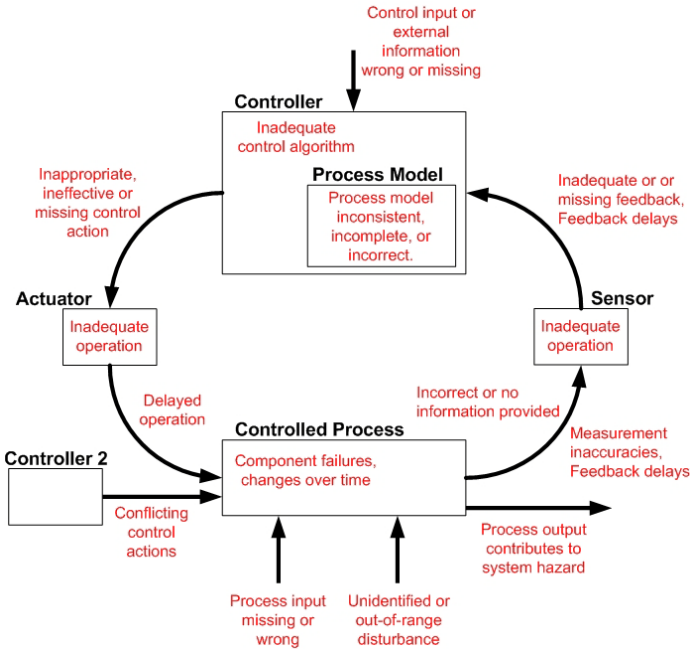

Figure 1 — The Causal Factors to be Considered in Step 2

Customizing STPA to Analyze the Tool Integration of an Industrial Tool Chain

Our objective for using STPA was to analyze a tool chain from an industrial case study to find risks that link tool integration to safety. STPA allowed us to not only analyze a single instance of this tool chain, but also to update it twice and analyze how the risks changed. This was possible both when the changes were motivated by safety considerations and when they were motivated by other goals (such as improved efficiency) (ref. 1).

Customization of Step 1 of STPA: The first action when customizing STPA to our context was to define the hazard we were analyzing as *the hazard that tool integration contributes to the introduction of hazards in the end product.* Based on this hazard we identified the implications of the basic (and additional) risks in regard to all possible *tool integration aspects*[10]. This allowed us to customize STPA by translating the basic risks into general types of risks for tool integration. Table 1 shows an excerpt of these translations, while the full details are found in reference 1.

Table 1 — General Types of Risks for Tool Integration

	(1) A control action …	**(2) An unsafe control …**	**(3) A potentially safe control …**	**(4) A control action required …**
Control Integration	A tool cannot provide an existing service to another tool.	The Control Integration Software (CIS) invokes the wrong service or a tool can provide an unqualified service to another tool.	Timing is affected by the CIS or missing parts of the CIS (manual invocation), affecting when a service is invoked.	Timing is affected by the CIS or missing parts of the CIS (manual invocation), affecting the invocation of a service.
Control to Platform Integration Relationship	No common link exists to provide an existing service from a tool to another.	The Platform Integration Software (PIS) invokes the wrong service or a tool relies on an unqualified platform service to provide a service to another tool.	Timing is affected by the PIS or missing parts of the PIS (manual invocation), affecting when a service is invoked.	Timing is affected by the PIS or missing parts of the PIS (manual invocation), affecting the invocation of a service.

[10] The five tool integration aspects are control, data, platform, presentation and process integration (ref. 2).

Based on these general types of risk, we could identify the programmatic risks of the case study. The identified risks are too many to enumerate in this paper (reference 1 has the complete lists), but we give three examples in Table 2.

Table 2 — Programmatic Risks and Causes

Risk	**Explanation**	**Causes**
(5) PRS and/or TS not correctly reproduced.	A Product Manager prepares a Product Requirement Specification (PRS) and Technical Specification (TS). These are later read by a System Architect who extracts relevant information and puts it into other development artifacts (like the System Design Specification (SDS)).	(a) System Architect does not understand PRS and/or TS (lack of training, tight time plan, etc.). (b) System Architect enters erroneous information in the SDS (large amount of information in unstructured text, etc.). (c) Systems Architect enters erroneously structured information in the SDS (wrong format, wrong table style, etc.). (d) SDS is corrupted during storage. (e) System Architect does not understand end product environment completely (unknown EMI, etc.). (f) No feedback on SDS from later development phases.
(41) The PRS and the TS contradict each other.	See above.	(a) No clear directive on how the PRS and TS should interrelate. (b) Product Manager does not understand the implications of the interrelationships between the PRS and the TS (lack of training, tight time plan, etc.). (c) PRS and TS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the different artifacts. (d) Different people review the different artifacts (both directly and through creating new documents based on the PRS and TS). (e) No common feedback given to the Product Manager on both the PRS and the TS.
(80) To create the FDS takes too long time, due to manual transformation of data. An incomplete version of the FDS is therefore used.	The FPGA Application Engineer reads through documents prepared by a System Architect (the FPGA Requirement Specification (FRS) and the Control Hardware Specification (CHS)) and writes the FPGA Design Specification (FDS). The FDS is later used by a Matlab/Simulink Application Designer to create other development artifacts.	(a) No clear directive on when the FDS is acceptable. (b) FPGA Application Engineer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). (c) FDS is not structured efficiently and/or appropriately. (d) Management Team enforces the acceptance of the FDS prematurely due to project timing constraints. (e) FRS and/or CHS are not structured efficiently and/or appropriately. (f) Late changes in the FRS and/or CHS require extensive rework of the FDS. (g) Late feedback on the FDS from the Matlab/Simulink Application Designer requires extensive rework of the FDS.

Customization of Step 2 of STPA: The use of the causal factors seen in Figure 1 is, as for the application of the basic risks in step 1, straight-forward for limited parts of the analysis when the context is suitable. However, each programmatic risk can belong to several control loops, of which some are context-specific and not directly obvious. To identify any such control loops for our context we used a conceptual model of a development effort (Figure 2) that we had defined during the construction of the control structure in step 1. This model consists of four different levels of organization. At the top, *management level*, we find the control and oversight of the effort, such as the procedures to establish a safety culture, certification agencies, project management, etc. One level below, at the *operator level*, we find the operators who are involved in the process towards producing end products, such as managers, developers, tool chain developers, etc. The next level, the *tool chain level*, consists of the tool chain used

during development and the bottom level, the *tool level*, of tools and supporting software used during development (such as tool integration mechanisms).

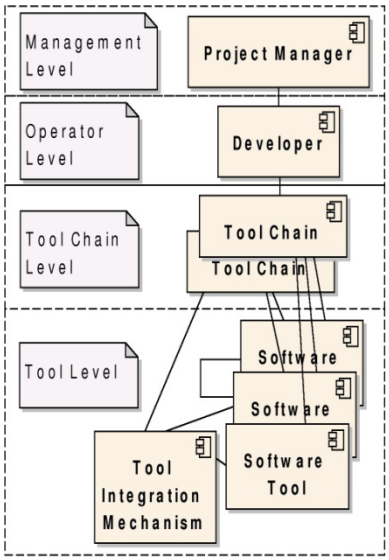

Figure 2 — Conceptual Model of a Development Effort

Of these levels, the tool chain level is specific to our context in that it considers the tools and tool integration mechanisms of a development environment as an integrated whole. The second action to customize STPA to our context was therefore to identify a control loop that (1) spans all operators and tool integration mechanism throughout the whole development process and (2) supports analyzing the hazard that tool integration contributes to the introduction of hazards in the end product. This was achieved by defining a control loop in which operators and tool integration mechanisms (the controllers) can introduce hazards in the end product (the controlled process) through development artifacts (the actuators), hazards which can be identified through verification and validation activities (the sensors).

Benefits of Applying STPA

Domain experts and developers using the industrial tool chain on a daily basis had identified 25 weaknesses previous to our analysis (ref. 1). 23 of these weaknesses were relevant to safety and 16 of these were identified by our analysis (the 7 that were not identified are discussed in the *Difficulties with Applying STPA* section). In addition, the 16 identified weaknesses were categorized by STPA into either risks or causes (Table 3 gives three examples).

Table 3 — Categorization of Expert Knowledge

Weakness Identified by Experts	**Identification through STPA**
Requirement engineering is poor. There is no traceability to implementation and back.	For instance identified through cause (f) to risk (5) (Table 2).
Requirement engineering is poor. The textual representation of the existing requirements may lead to different opinions on the meaning.	For instance identified directly through risk (41) (Table 2).
Too much FPGA rework. Workflow is based on manually created specification documents.	For instance identified directly through risk (80) (Table 2).

The obvious benefit of STPA in comparison to relying on expert knowledge is the systematic categorization *into* risks and causes. This is a major benefit when performing safety-guided design, since you can guide redesign efforts by (1) the possibility that the risks can occur and (2) what needs to be controlled (the causes). However, this is not

unique to STPA, but rather fundamental to system safety in general. The major benefit in our context was the support by STPA *in categorizing* risks and causes.

First, the basic risks, translated into general types of risks, identified two categories of risks, namely one static, binary category and one dynamic category. The risks pertaining to the former category could be mitigated prior to the use of a tool chain, since they relate to static circumstances. An example of this would be risk (5) in Table 2. Either the System Architect performs this action, or some software component does. Either there is some protection of data consistency, or there is not. The latter category contained risks and causes that could occur due to timing issues, as for example risk (80) and its causes (d), (f) and (g) in Table 2. These risks can be planned for, but to verify that they have been mitigated is difficult prior to the use of the tool chain. This categorization is critical in the context of tool integration, since there are typically different *stakeholders* involved before and after the deployment of a tool chain and therefore very different possibilities for mitigating the different categories.

Secondly, the causes were related to different stakeholders depending on which part of which control loops they pertained to. See for instance risk (80) in Table 2. In regard to the customized control loop described in the previous section cause (a) relates to the control input (relevant stakeholder at the management level), cause (b) relates to the controller (the FPGA Application Engineer at the operator level) and the causes (c), (e) and (f) relates to the actuators (the responsible for different development artifacts at the tool level). The causes (d) and (g) relate to the sensors of a local process control loop (Management Team and Matlab/Simulink Application Designer at the operator level). The possibilities for mitigating the causes are typically tied to the stakeholder responsible for the inadequate control and the identification of these stakeholders was significantly easier with the guidance provided by STPA.

Difficulties with Applying STPA

When analyzing the industrial tool chain we came across three major difficulties, namely identifying basic risks for new contexts, defining control structures and limiting the domains (management, user, technical, etc.) taken into account.

Identifying context-specific causal factors: As mentioned in the *Customizing STPA to Analyze the Tool Integration of an Industrial Tool Chain* section, STPA allows for the definition and use of additional risks based on the nature of the context studied. An example of an additional risk that we considered is the risk that the verification and validation activities test the wrong behavior of the end product, i.e. that the correct sensors do not exist. This is common when there are many or complicated requirements. We eventually decided that it would be more straight-forward to cover this during the step 2 of STPA by defining the "inadequate operation" of a sensor to also cover this eventuality. However, while keeping this in mind is straight-forward for an expert on tool integration, it would be difficult for the ordinary user (i.e. an analyst who is an expert in the tool chain at hand, rather than tool integration in general).

The use of guidewords to further guide the search for causes when using STPA has been suggested for the context of humans and organizations (ref. 17). A similar approach is needed to allow the ordinary user to analyze an embedded systems development effort, but further research is needed to identify the appropriate guidewords. A start has however been identified in 9 safety-critical properties of tool chains (Data Integrity, Data Semantics, Tool Automation, etc.) identified in reference 1.

Defining control structures: A critical part of STPA is the definition of the control structure during step 1, i.e. to define all relevant system components and their relationships. In our context we defined the control structures as descriptions of the development process used during the development effort (ref. 1). When performing step 1 of STPA the control structures can be seen as an advantage in that they create a finite scope for the analysis. This view is for instance advocated in reference 14, which presents a safety assessment methodology based on STPA. An assumption behind this view is that the control structure is being defined by limiting the available information (reference 14 mentions interface specifications, design documentation, user manuals and user interface designs as sources for extracting information). However, during safety-guided design the control structure will evolve based on the hazards identified and rather require the constant expansion of the gathered information. This is especially difficult as new domains become relevant (for instance when the safety culture of the development environment can no longer be ignored and requires an analysis of the whole management level).

This difficulty became apparent during our analyses in two ways. First, most of the 7 relevant weaknesses in our tool chain that we did not identify when using STPA were related to missing or not well defined control loops. That the process is unknown or -to some extent- arbitrary is not uncommon in development efforts. There is, however, no support by STPA as such for identifying missing parts of or simplifying control structures, even though it should come as no surprise that defining the system is often the most difficult part of an analysis (ref. 8). STPA instead seems to rely heavily on the engineering expertise of the user. Secondly, when mitigating certain risks, new risks can be created in other domains. An example is a risk similar to risk (80) in Table 2, which was solved by automation. This required the engineering expertise known by the involved engineer to be transferred to the tool chain developer involved in developing the automated data transfer. The new risk, associated with the organizations rather than the engineer, was that this would not be done correctly. It would therefore seem as developing control structures for all domains as they become relevant does not only require engineering expertise, it requires *diverse* engineering expertise.

Accident analysis can be enhanced when performed by diverse teams (ref. 17) and this diversity is critical for defining control structures during safety-guided design. To be fair to STPA it can however be noted that the activity of creating a control structure in itself can lead to much needed discussion and not adding some components to the system model is sometimes also a conscious choice (we did for instance choose not take governmental or legal issues into consideration in reference 1).

Limiting the domains taken into account: The last identified difficulty relates to which domains to take into account during the analysis. 13 operators interacted with our industrial tool chain and all of them belong to different work categories (ref. 1) (software developers, project managers, etc.). The work practices of these work categories are different, since the particulars of their activities are very different. A software engineer is for instance freer to experiment when producing development artifacts, than a hardware engineer who is limited by long manufacturing times. By knowing the work practices of an engineering discipline one can better judge the possibility that the causes identified in step 2 of STPA will occur. The gain of using a more granular definition of work categories (for instance by partitioning a “developer” category based on engineering disciplines) would therefore be a more accurate understanding of the possibility of a risk occurring. Whether to define and include these differences in the analysis therefore depends on the acceptable scope (in regard to resources, time, etc.) of the analysis contrasted against the value of a better prioritization of risks.

Although we were not limited by any restrictions on the scope of the analysis we anyway opted for a generic approach, since we also did not need to prioritize between risks. To keep the analysis on the correct level we continuously referred to the conceptual model described in the *Customizing STPA to Analyze the Tool Integration of an Industrial Tool Chain* section (Figure 2). In fact, to continuously keep in mind the different domains that were to be taken into account seemed to be critical to handle the amount of details when applying STPA (i.e. to avoid looking in too much detail on the technology when analyzing the activities of the operators).

Improvements

Based on the previous section we can identify two urgent needs in relation to supporting the application of STPA.

Expert Systems: A software program containing knowledge on how control loops pertaining to different domains (management, user, technical, etc.) could be constructed and how they might fail would be of great help. It would be especially helpful if this system allowed for constructing and combining control structures from different domains (for instance combining an organizational structure with the handling of a complex technical system through the operators involved). Besides the obvious advantage of disseminating knowledge from domain experts to STPA users such a system could assist in (1) differentiating between weak and strong parts of control structures, (2) the identification of faults that occur due to the inadequate, simultaneous control in several control loops, (3) predicting the time needed to finish a STPA safety analysis based on the complexity of control structures, (4) identification of improvements that come at a low or no cost (such as when enforcing safety constraints in a specific way can allow one to remove other safety measures), (5) indicating the completeness of control structures, etc.

To our knowledge there is no expert system available that support the aspects of STPA mentioned in the previous paragraph. What is currently available is a number of publications that describe the application of STPA or STAMP in different domains. Reference 14 describes the application of STPA on a Fictional Missile Interceptor System.

Reference 12 details a STAMP analysis of the Comair 5191 aircraft accident in 2006. Reference 13 uses STPA to analyze the China–Jijiao railway accident in 2008. Reference 7 gives an overview of the definition of a control structure for infection outbreaks in hospitals. Reference 10 gives examples of applying STAMP and STPA in several domains, including cases related to space and public health.

Simulation: Simulations through which users can explore the consequences of inadequate control in different parts of a control structure would be helpful in assisting STPA users to understand the system they are working with and choosing appropriate strategies for updating the system. It would also support the analysis of dynamic processes, such as when fast and slow control loops lead to asynchronous evolution or inconsistencies between mental models.

Reference 9 refers to the use of the SpecTRM-RL modeling language to create executable and analyzable models. However, this language seems mostly suited for modeling physical systems. Reference 5, 7, 9 and 17 all advocate the use of System Dynamics Modeling (ref. 10) to model and simulate the control structure of the development and operation of complex socio-technical systems. The support in analyzing how the control structures change over time is emphasized as especially important. Reference 5 describes an especially detailed investigation on "a new framework to create dynamic executable models used to analyze time-dependent risks, assist engineers and managers in safety-related decision-making, create and test risk mitigation actions and policies, and monitor the system for states of increasing risk." However, the creation of commercial quality software, without which extensive expert knowledge on the modeling and simulation techniques is needed, is specifically mentioned as important *future* work.

Conclusions

STPA is well suited for analyzing the safety issues related to contexts in which a large number of stakeholders cooperate in implicit ways, such as in a tool integration context in which different stakeholders develop, procure, deploy, configure and use the development effort at hand. However, to be effective during safety-guided design STPA does not only require a deep knowledge in the analyzed development effort, but also in tool integration, human psychology, organization studies, etc. This will create difficulties for the ordinary user of STPA, especially in relation to identifying context-specific causal factors, defining control structures and deciding which domains are important enough to consider. In other words, it will be difficult for the ordinary user of STPA to judge the completeness of the analysis and whether a change indicated by the analysis is going to have side effects. These difficulties ultimately point to the need for expert systems targeted at modeling and simulating control structures during STPA analyses.

References

1. Asplund, Fredrik. *Safety and Tool Integration, A System-Theoretic Process Analysis, TRITA-MMK 2012:01, ISSN 1400-1179, ISRN/KTH/MMK-R-12/01-SE.* Industrial Case Study, Stockholm: KTH, 2012.
2. Asplund et al. "Tool integration beyond Wasserman, Advanced Information Systems Engineering Workshops: CAiSE 2011 International Workshops." London, UK, 2011.
3. Asplund et al. "Tool Integration, from Tool to Tool Chain with ISO26262." Society of Automotive Engineers World Congress, 2012.
4. British Standards Institution on ERC Specs and Standards. *BS IEC 61511-3:2003 - Functional safety - Safety instrumented systems for the process industry sector - Part 3: Guidance for the determination of the required safety integrity levels.* 2003.
5. Dulac, Nicolas. *A Framework for Dynamic Safety and Risk Management Modeling in Complex Engineering Systems.* Ph.D. Thesis, Boston: MIT, 2007.
6. International Organization for Standardization. *ISO 26262 - Road vehicles - Functional safety.* 2011.
7. Waterson, Patrick, and P. W.H. Chung. "Investigating the Potential to Simulate Hospital-Based Infection Outbreaks Using The STAMP Architecture." *The 5th IET International System Safety Conference 2010.* Manchester, UK: Loughborough University, 2010.
8. Leplat, Jacques. "Occupational Accident Research and Systems Approach." *Journal of Occupational Accidents*, 1984: 77-89.
9. Leveson, Nancy. "A New Approach to Hazard Analysis for Complex Systems." *Proceedings of the 21st International System Safety Conference,*. Ottawa, Ontario Canada: System Safety Society, 2003. 498–507.
10. Leveson, Nancy. *Engineering a Safer World.* The MIT Press, 2011.

11. Leveson, Nancy. *The Use of Safety Cases in Certification and Regulation.* ESD Working Paper Series, Boston: MIT, 2011.
12. Nelson, Paul S. *A STAMP Analysis of the LEX Comair 5191 Accident.* Master Thesis, Lund: Lund University, 2008.
13. Ouyang, Min, Liu Hong, Ming-Hui Yu, and Qi Fei. "STAMP-based analysis on the railway accident and accident spreading: Taking the China–Jiaoji railway accident for example." *Safety Science 48*, 2010: 544–555.
14. Pereira, Steven J., Grady Lee, and Jeffrey Howard. "A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System." *Proceedings of the 2006 AIAA Missile Sciences Conference.* Monterey, CA, 2006.
15. Special Committee 205 of RTCA, Inc. *DO-178C, Software Considerations in Airborne Systems and Equipment Certification.* RTCA, Inc, 1992.
16. Storey, Neil. *Safety-Critical Computer Systems.* Harlow: Addison Wesley Longman, 1996.
17. Stringfellow, Margaret V. *Accident Analysis and Hazard Analysis for Human and Organizational Factors.* PhD Thesis, Boston: MIT, 2010.

Biography

Fredrik Asplund, M.S., Ph.D. Candidate, KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Brinellvägen 83, 10044 Stockholm, Sweden, telephone – +46 (0)8 790 74 05, e-mail – fasplund@kth.se.

Fredrik Asplund has been studying towards a Ph.D. at the KTH Royal Institute of Technology for two years, partly focusing on System Safety. Prior to this he worked for 7 years in the telecommunications and automotive industries as a developer, software architect and team lead. Fredrik is the main responsible for the research leading up to the presentation. He gives presentations at regular intervals, as a teacher and at research conferences.

Jad El-khoury, Ph.D., Researcher, KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Brinellvägen 83, 10044 Stockholm, Sweden, telephone – +46 (0)8 790 68 77, e-mail – jad@kth.se.

Dr. El-khoury is a researcher in the area of embedded system development, with experience in model-based development and tool integration.

Martin Törngren, Professor, KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Brinellvägen 83, 10044 Stockholm, Sweden, telephone – +46 (0)8 790 63 07, e-mail – martint@kth.se.

Martin Törngren is Professor in Embedded Control Systems at KTH where his research group is addressing architectures and methodologies for embedded control systems development. Martin is the principal initiator and director of the KTH Centre for Embedded Systems.