Safety and Tool Integration, A System-Theoretic Process Analysis

FREDRIK ASPLUND
KTH ROYAL INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MACHINE DESIGN
DIVISION OF MECHATRONICS

KTH Industriell teknik
och management

TRITA–MMK 2012:01
ISSN 1400-1179
ISRN/KTH/MMK-R-12/01-SE

Technical Report
Stockholm, Sweden 2012

ABSTRACT

In this report I detail a *System-Theoretic Process Analysis* (STPA) hazard analysis of the tool integration of development environments for embedded systems. Building on results from previous studies I generalize and expand on earlier findings regarding the relationship between safety and tool integration.

To prepare for the analysis I customized STPA for the context of tool integration. This customization allowed me to subsequently design and analyze three versions of a tool chain originally provided by an industrial partner. A net result of 85, 98 and 73 risks was identified, in comparison to 25 integration weaknesses identified through expert knowledge. The design of the different versions of the tool chain and a comparison of the identified risks with the integration weaknesses allowed me to validate the usefulness of STPA for both identifying and correctly categorizing risks and causes in the context of tool integration. An analysis of my results also points out the fact that STPA is not a silver bullet, without enough expertise it is easy to omit important parts of process models and thus arrive at incomplete conclusions.

In regard to the relationship between safety and tool integration nine properties were identified, properties that need to be supported correctly to avoid hazards in the context of tool integration. These properties require support throughout a noticeable part of a development environment to have an impact and derive much of that impact from the possibility to centralize them. They also interrelate, so that often several of them need to be handled to mitigate one type of risk. However, introducing support for them across a whole development environment is likely to be costly, or even impossible. Furthermore, introducing support for these properties will mitigate some risks, but also create other risks at higher levels of organization.

These properties therefore point to the size a development environment, the number of contexts towards which the development environment can be verified and the effort required to ensure the added requirements at higher levels of organization as deciding factors on whether the effort to support them should be made (other efforts, more efficient in those particular cases, could otherwise be considered). The existence of these properties also point to the possibility of developing and pre-qualifying tools and tool chains based on the assumption that some or all of these properties will be supported by the final development environment. This could potentially lower, or at least distribute, the cost of the final qualification.

1 INTRODUCTION

iFEST[1] is an Artemis research project aimed at providing an Integration Framework for tool chain conception and maintenance within the engineering domain of complex industrial embedded systems. As part of this effort several industrial case studies are being built for use as demonstrators and means of evaluating the iFEST Integration Framework. These case studies also provide contemporary examples for analyses related to embedded systems and tool integration.

In [1] I and my colleagues detailed an initial effort to clarify the relationship between safety and tool integration. We were also able to identify several characteristics of tool chains that, when safety is a concern, could help guide tool integration. However, the conclusions were reached by applying the newly released ISO 26262 to a relevant case study. This approach meant that the results were restricted to the domain[2] of ISO 26262. To progress further a more generic approach is needed, such as an exhaustive search for hazards in the context of a case study on embedded systems development.

This report details a *Hazard Analysis Technique* (HAT) analysis of one of the iFEST industrial case studies and a subsequent analysis of the impact on end product safety due to tool integration (and vice versa). The case study is presented in an anonymized form, but retains enough detail to allow relevant reasoning.

Chapter 2 of this report provides an introduction to the field of tool integration and relates it to the safety of an end product. Based on Chapter 2, chapter 3 describes how the first step of the *System-Theoretic Process Analysis* (STPA) [2] HAT can be customized to allow analysis of end product safety as related to tool integration (and vice versa). Chapter 4 then provides a brief description of the background to the case study, while chapter 5, 6 and 7 goes into detail regarding the three different tool chains analyzed in the case study and my STPA of them. These tool chains are the original tool chain used by our industrial partner and two subsequent iterations of it (still supporting the development of the same end products). In chapter 8 I continue customizing STPA (this time the first part of the second step) and analyze the tool chains further. Chapter 9 contains a discussion of the results, which is then summarized in the conclusions in chapter 10.

[1] http://www.artemis-ifest.eu/

[2] The application sector of Electrical and/or Electronic (E/E) systems within road vehicles [6].

2 THE CONTEXT OF TOOL INTEGRATION

To develop an embedded system there is typically a need for a multitude of *engineering tools*, i.e. tools that provide functionality to fulfill one or more activities without which the finished product cannot be cost-efficiently developed. Examples of engineering tools range from design tools to test tools, from requirements tools to production configuration tools, etc.

These engineering tools execute in the context of a *development environment*, defined as all the other tools and any supporting software used during development. An important aspect of a development environment is that the *development processes* it supports will define orderings of the engineering tools it contains, i.e. *tool chains*, which the development of a product will transit through. *Tool integration* can be defined as that which supports the development process in correctly moving from one engineering tool to another in a tool chain. Examples of tool integration range from definitions of transformations of data between different data formats to transformation tools, from scripts that react to the actions of a user to process engines, etc. When discussing tool chains, tool integration software is usually implicitly included as something needed "behind-the-scenes" to enable the transition between engineering tools.

Tool integration can also be a manual process, an example being when a software developer reads a number of documents provided by software architects and proceeds to program a software component. In this example the data provided in an informal language as output from a word processor has been translated to code which (presumably) a large number of developer tools can operate on (static analyzers, compilers, etc.). Nevertheless, when one speaks of "an increase in tool integration" or "a more integrated development environment" one is traditionally referring to an increase in or expanded functionality of tool integration software.

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 [3] even though these are only the end result of striving to handle complexity, provide tracing, etc., to support the primary goals of ensuring end product safety, cost-efficiency, etc. In [4] I and my colleagues introduced a reference model for reasoning about tool integration at an abstract level, to which non-functional properties are of central importance.[3]

The two subsequent subchapters describe, summarize and build upon the findings from [4] and [1] in preparation for the analyses found later in the report.

2.1 THE REFERENCE MODEL

Reasoning about tool integration is supported by our reference model by dividing tool integration into *control*, *data*, *platform*, *presentation* and *process integration aspects* and defining *dependencies* and *metrics* for these (see **Figure 1**).

[3] A note to the reader, in this text "our reference model" is therefore a reference to the reference model developed by me and my colleagues, while "our context" is a reference to the context dealt with by this report (i.e. tool integration as applied to the development of embedded systems).

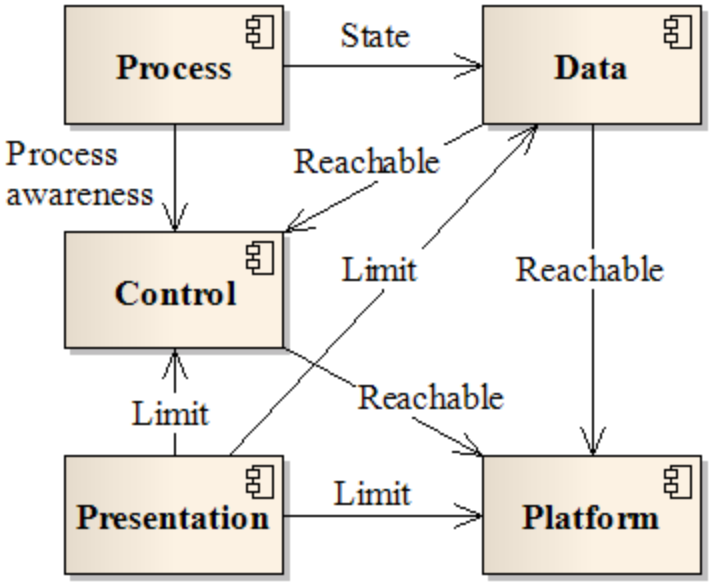

FIGURE 1 THE INTEGRATION ASPECTS

- 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 separate 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.

To exemplify the tool integration aspects and their relationships picture two development activities; the first involves a hardware engineer designing the hardware interface of a *Field-Programmable Gate Array* (FPGA), while the second involves a software developer designing software that uses the hardware interface to enable a safety-critical functionality handled by the FPGA. There are many ways of handling this implicit relationship during development, for instance one may envision a process that requires the software developer to review and give feedback on the hardware interface to the hardware engineer.

To support the review process the choice may therefore be made that both the software developer and the hardware engineer should work with software tools running on operating

systems (OSs) that support the most common networking protocols. If the computers of both the software developer and the hardware engineer are then connected to an intranet a common environment has been established through *platform integration*. The OSs make up the *Platform Integration Software* (PIS).

Building on the platform integration the tool chain developers may create services that the software developer may use to retrieve the hardware interface files for review, i.e. *control integration*. The *Control Integration Software* (CIS) could in this case be made up of parts of the OSs (perhaps they provide a virtual file system) or additional, separate software (for instance a Version Control System). The relationship between platform and control integration is readily identifiable, if the software developer and the hardware engineer work on platforms which are not linked to each other then they cannot access services on each other´s platform. Furthermore, the relationships between platform, control and data integration can also be distinguished in this example; if the platforms are not linked **or** if there are no services available to request the relevant files, then data cannot be shared between the software developer and the hardware engineer.

In many cases a software developer would not be proficient in handling a software tool for hardware design. Tool chain developers may then introduce a transformation engine that is able to transform the hardware interface files into C source code files, i.e. *data integration*. The *Data Integration Software* (DIS) is then made up of the transformation engine and any associated software (for configuration, verification, etc.).

Another way of solving the problem mentioned in the previous paragraph would be to build a new GUI, separate from the software tools for hardware design and tailored to the needs of the software developer, for reviewing the hardware interface files. This would be *presentation integration* and the new GUI *Presentation Integration Software* (PreIS). The relationships between presentation integration and other integration aspects can also be exemplified here. The graphical toolbox (or similar) of the available platform will limit how the software developer can perceive the customized GUI; the available services will limit how the software developer can interact with the development environment through the customized GUI; the semantics and syntax of the available data will limit what information the software developer can gain from the customized GUI.

Finally, tool chain developers may not trust the software developer to pick the correct version of the hardware interface files, perform all the required steps to transform the data correctly, etc. They may then choose to introduce scripts that trigger transformations upon each new hardware interface file that is submitted, i.e. *process integration* aimed at integrating the process into tool integration mechanisms[4]. The scripts are *Process Integration Software* (ProIS), and depending on the setup other software may also fall into this category (for instance a tool that monitors the development environment and triggers the scripts at appropriate times). The relationship between process integration and control integration is the event that triggers the process logic; the relationship between process integration and data integration is the possibility of the process logic to determine the state of the process (i.e. which the current and next relevant activities are).

The difference between the relationships of presentation integration and the relationships of process integration may seem odd at a first glance; is not the process logic dependent on the development environment in a similar way as the average operator is? To understand the difference one must remember what is actually being integrated. In the case of presentation

[4] We call the smallest part of tool integration software that provides a distinct functionality a *tool integration mechanism*.

integration it is the environment of the individual operator that is being integrated to enable a clear perception of, a straight-forward interaction with and a correct understanding of the development environment. The target is to enable the unpredictable interaction between operator and development environment, which may occur solely due to experimentation. In the case of process integration it is the overall process that is being integrated, akin to integrating the actions of all operators into the development environment as rigid, inflexible rules. This means that the relationship with the platform is much more unclear, as these rules doesn´t have to be centralized (the process could be choreographed based on for instance time, or rely on operators starting work after an artifact on another platform has been approved). The emphasis in our reference model is therefore, for now at least, the possibility of process logic to ensure that it is acting in the right context (neglecting the active process actions that are limited in what they can achieve through the control and platform integration aspects).

2.2 SOFTWARE QUALIFICATION FOR SAFETY

Standards dealing with safety can either take a primarily *prescriptive* ([5], etc.) or *descriptive* ([6], [7], [8], etc.) approach on how to enforce safety. While the prescriptive standards focus on giving an exhaustive list of exactly how a safety-critical product should be built and tested, the descriptive approach points out requirements on appropriate environments, methods and processes for developing safety-critical products. The latter of these types imply requirements and guidelines on how to qualify software, while the former implies lists of approved software tools, integration frameworks, etc.

Generic guidelines for qualifying software tools can be found in ISO 26262 [6] (and, more or less, in related standards). The aim of these guidelines is to ensure a sufficiently low possibility of safety requirement violations due to the use of a tool. This requires a low possibility of violations being caused by erroneous output from the tool and/or the tool failing to detect errors introduced by it or other tools. One can deduce that, in the context of software tools used during the development of safety-critical systems, a “safe” software tool is one that does not by itself contribute to the accumulation of defects that can violate safety constraints in the end product. This is a sufficiently generic statement to be applicable also to software used for tool integration purposes.

Therefore, if engineering tools add to the software qualification effort needed to ensure a safe development environment, so should tool integration software. In modern development environment, which may contain thousands of tools [9] and related tool integration mechanisms, this unfortunately implies a tremendous qualification effort.

The question is then whether there are characteristics of tool chains that lower the possibility of safety requirement violations, since there are characteristics of engineering tools that lower the possibility of safety requirement violations (for instance internal logic verification, such as a compilers internal verification of the logic of the generated code). Such characteristics of tool chains may allow focusing a qualification effort to the benefit of both safety and cost-efficiency. An initial effort, presented in [1], suggests that such characteristics do in fact exist. Those identified were *process control* that ensures the safety of lifecycle process and *traceability* that ensures completeness and consistency.

In the next chapter I describe how the first step of the STPA HAT can be customized to allow analysis of end product safety as related to tool integration. This is a first step in expanding and generalizing the results from [1].

3 HAZARD ANALYSIS IN RELATION TO TOOL INTEGRATION

As mentioned in chapter 1, the conclusions in [1] were reached by analyzing the compiled knowledge of safety that the newly released ISO 26262 represents, as applied to a relevant case study involving tool integration. To expand on this requires a generic analysis of what *hazards* are likely to ensue from the use of tool integration in an embedded development project. There are several different HATs available to support the search for hazards.

The first question then becomes which HAT is the most appropriate in this case. *Failure Modes and Effects Analysis* (FMEA) and *Fault Tree Analysis* (FTA) are two widely used HATs that could be considered [10]. However, both FMEA and FTA focus firmly on distinct failures and how these propagate through a system, which is a limitation when the system is complex and abstract (which is the case for the analyzed industrial case study). Hazard and Operability Study (HAZOP) is another widely used HAT [11] which 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. In the context of embedded systems development the dependencies between parts of a development environment 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 can be supplemented with other HATs such as Layer of Protection Analysis (LOPA) [12] to better handle some of these issues (LOPA, for instance, is suited for analyzing hazard mitigation mechanisms pertaining to different domains), but the approach of combining several HATs would require a larger effort.

Ideally one would want to use a single HAT that combined the benefits of providing guidance and the efficient handling of different domains. In this specific case it should also support the possibility of use before a system has been built. STPA is a HAT that is designed with the first two benefits in mind and allows for use already during design. Due to these benefits I chose STPA for my analysis.

To my knowledge there does not exist any documented effort to apply STPA in the context of tool integration. So, to be able to use STPA I must therefore first make an effort to customize STPA according to the context of tool integration and then, after the analysis, make sure to analyze whether this HAT was in fact suitable to this context.

3.1 STPA IN THE CONTEXT OF TOOL INTEGRATION - CUSTOMIZED

The first step of a STPA is to [2]:

“Identify the potential for inadequate control of the system that could lead to a hazardous state. Hazardous states result from inadequate control or enforcement of the safety constraints, which can occur because:

- A control action required for safety is *not* provided or not followed.
- An unsafe control action *is* provided.
- A potentially safe control action is provided too early or too late, that is, at the wrong time or in the wrong sequence.
- A control action required for safety is stopped too soon or applied too long.”

This is straight-forward at lower levels of organization[5], at which the inadequate control is performed by a physical end product (such as a valve, microcontroller, etc.). STPA is however also applicable at higher levels of abstraction, but then requires:

- An effort to translate the basic possibilities for inadequate control to an appropriate form. An example is given in [2] for the organizational and management components in the context of NASA. In this example analyzers translated the basic possibilities for inadequate control for the hazard that an accident could take place due to unsafe decision-making. The result of the translation was:
    - (1) Unsafe decisions are made or approved.
    - (2) Safe decisions are disallowed.
    - (3) Decision making takes too long.
    - (4) Good decisions fail to impact the system design, construction and operation adequately.
- That one takes additional risks due to the nature of the higher levels of organization into consideration. For organizational components [2] coordination risks are listed as additional risks, i.e. that "(1) both controllers assume the other is performing the control responsibilities and as a result nobody does or (2) controllers provide conflicting control actions that have unintended side effects."

Table 1 lists my translation of the basic possibilities for inadequate control for the hazard that tool integration contributes or leads to an unsafe product being produced. This translation has been done for each tool integration aspect and relationship between tool integration aspects in our reference model. I have then done the same for coordination risks.

Based on **table 1** I can summarize the general types of risks within our context as:

- Safety-critical[6] services cannot be provided, due to errors or missing parts in the CIS, PIS or Tool Interfaces.
- Erroneous safety-critical services are provided, due to errors in the CIS, PIS or Software Tool Qualification.
- Safety-critical data cannot be interpreted, due missing parts in the DIS.
- Safety-critical data is corrupted, due to errors in or missing parts of the DIS, CIS, PIS or Software Tool Qualification.
- Safety-critical data is not provided, due to errors in the DIS.
- Safety-critical data is not possible to request / update, due to missing parts in the PIS, CIS or Tool Interfaces.
- A user achieve an unintended result, due to being presented with an, to his context, unfamiliar UI.
- That the state of the safety-critical data is insufficient for a specific process activity is not detected, due to erroneous or lacking possibilities to analyze the data.
- Safety-critical services are invoked too early or too late, due to timing issues in or missing parts of the CIS or PIS.
- Safety-critical services are stopped too soon or applied too long, due to timing issues in or missing parts of the CIS or PIS.

[5] In *systems theory* a complex system can be modeled as a hierarchy of levels of organization, each level imposing constraints on the degree of freedom of the components at lower levels (see [2]). In this report we refer to the different levels of organization when modeling a development effort, for instance the operator level, the management level, etc.

[6] With safety-critical in this context I am referring to data, or services that deal with data, that relates to parts of the end product that are safety-critical.

- Safety-critical data is not complete when provided, due to timing issues in or missing parts of the CIS, PIS or DIS.
- Safety-critical data is provided too early or too late, due to timing issues in or missing parts of the CIS, PIS or DIS.
- Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.

Of these the first eight are relevant to the preparation prior to using a tool chain[7]; the subsequent four are relevant when the tool chain is actually used; the last general type of risk is relevant both when preparing and when using a tool chain.

The next four chapters describe the background and details of the iFEST industrial case study which I use the customized version of STPA to analyze. The potential for inadequate control of the system that could lead to a hazardous state (which is to be identified in Step 1 of STPA) is presented as enumerations of *Programmatic Risks*[8] in tables referenced in these chapters.

[7] In the sense that these eight risks can be identified and solved prior to the use of a tool chain.
[8] The Programmatic Risks are the specific unsafe controls (in a specific context) in a safety control structure that can lead to the relevant general risks.

	A control action required for safety is not provided or not followed	An unsafe control action is provided	A potentially safe control action is provided too early or too late	A control action required for safety is stopped too soon or applied too long	Coordination Risks, i.e. that "(1) both controllers assume the other is performing the control responsibilities and as a result nobody does or (2) controllers provide conflicting control actions that have unintended side effects."
Control Integration	A tool cannot provide an existing service to another tool	The 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	(1) The invocation of CIS fails to invoke all relevant services from the actual tool or (2) result in conflicting services being invoked from it
Control to Platform Integration Relationship	No common link exists to provide an existing service from a tool to another	The PIS invokes the wrong service or a tool relies on a 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	N/A
Data Integration	Data provided by a tool cannot be interpreted or is interpreted erroneously by another tool, since there is no possible, available or correct data transformation	A tool receives corrupt data (erroneous, wrong version, etc.) due to errors in data transformation	Timing is affected by the DIS or missing parts of the DIS (manual data transfer), affecting when data is provided	Timing is affected by the DIS or missing parts of the DIS (manual data transfer), affecting the amount or completeness of the data provided	(1) Data input to the DIS is not delivered, since the correct source of data is unclear or (2) Conflicting data is provided from several sources.
Data to Control Integration Relationship	No service to request/update relevant data is provided by a tool, and data is corrupted during manual transfer	Corrupt data is provided/created, due to an unqualified service, erroneous CIS or missing CIS (manual transfer)	Timing is affected by the CIS or missing parts of the CIS (manual invocation), affecting when data is provided	Timing is affected by the CIS or missing parts of the CIS (manual invocation), affecting the amount or completeness of the data provided	N/A
Data to Platform Integration Relationship	No common link exists to request/update data, and data is corrupted during manual transfer	Corrupt data is provided/created, due to a unqualified platform service, erroneous PIS or missing PIS (manual transfer)	Timing is affected by the PIS or missing parts of the PIS (manual data transfer), affecting when data is provided	Timing is affected by the PIS or missing parts of the PIS (manual data transfer), affecting the amount or completeness of the data provided	N/A
Platform Integration	A platform service is not understood as safety critical and is not provided	A platform service is not understood as safety critical and is left unqualified	N/A	N/A	Insufficient platform integration leads to controllers not being able to communicate, and (1) or (2) occur
Presentation Integration	Providing a GUI that a user is not used to interact with leads to it not being understood, and no action is taken	Providing a GUI that a user is not used to interact with leads to it not being understood, and the wrong action is taken	Providing a GUI that a user is not used to interact with leads to it not being understood , and actions are taken at the wrong time or in the wrong sequence	Providing a GUI that a user is not used to interact with leads to it not being understood, and activities are stopped too soon or applied to long	Providing a GUI that a user is not used to interact with leads to it not being understood, and (1) or (2) occur
Presentation to Control, Data and Presentation Integration Relationships	Not possible to provide a GUI that a user is used to interact with, and no action is taken	Not possible to provide a GUI that a user is used to interact with, and the wrong action is taken	Not possible to provide a GUI that a user is used to interact with, and actions are taken at the wrong time or in the wrong sequence	Not possible to provide a GUI that a user is used to interact with, and activities are stopped too soon or applied to long	N/A
Process Integration	Process logic is misunderstood, erroneous or missing, and a required Process Activity (PA) does not occur	Process logic is misunderstood, erroneous or missing, and a erroneous PA occurs	Process logic is misunderstood, erroneous or missing, and PAs occur at the wrong time or in the wrong sequence	Process logic is misunderstood, erroneous or missing, and PAs are stopped too soon or applied to long	Process logic is misunderstood, erroneous or missing, and multiple process logic engines / triggers fail to activate or provide conflicting control over a required PA
Process to Control Integration Relationship	Not all services needed during a PA is invoked	A PA invokes (the) wrong / (a) unqualified service(s)	A PA invokes services at the wrong time or in the wrong sequence	A PA stop invoked services too soon or apply them too long	N/A
Process to Data Integration Relationship	The state of the data is not sufficient to ensure that the requirements of the process have been achieved, but it is used as if this was the case.	The state of the data regresses below process requirements for a process activity	N/A	N/A	The state of the data of different artifacts contradict each other, making the total state unknown

TABLE 1 SAFETY RISKS IN THE CONTEXT OF TOOL INTEGRATION

4 CASE STUDY DESCRIPTION

The case study is a tool chain from an industrial partner, which is developing embedded "Closed-loop Control System" products[9]. There are implications on safety if the control of the developed system acts in an unintended way[10] after product launch. Three different versions of the tool chain has been designed (see **Figure 2**, **3** and **4**), each one including more tool integration than the previous version. Each version of the tool chain is described in detail in one of the three chapters subsequent to this one. Besides the generic reasons stated in these chapters (i.e. improved efficiency and a need to mitigate some safety issues), there has been no specific requirements for the changes. The approach has been to increase the tool integration in the tool chain and then analyze the implications for safety due to these changes.

The following standards are adhered to or used as references when developing with the original tool chain: EN 50128:2001 (Traction applications with SIL 0) for Safety, IEC 60068 (General thermal conditions and type tests), IEC 60255-5 (Insulation test), IEC 61000-6-2 (EMC Immunity Generic Standard for industrial environment), IEC 61000-6-4 (EMC Emission Generic Standard for industrial environment), EN 50126 (Railways applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)), EN 50121 (EMC Immunity), TS 45545 NF F 16-101 & -102 (Fire Protection), EN 61373 (Vibration Levels Tractions) and IEC 60255 (Vibration tests) for environmental conditions for the controller hardware.

Despite assurances that these standards are used I assumed the worst case if no proof of the opposite could be found, i.e. I assumed that control mechanisms that were not explicitly stated were not present.

[9] In a closed-loop control system, a sensor monitors the output (i.e. a vehicle's speed) and feeds the data to the controller which continuously adjusts the control input (i.e. a throttle) as necessary to keep the control error to a minimum (that is, to maintain the desired speed). Feedback on how the system is actually performing allows the controller (i.e. a vehicle's on board computer) to dynamically compensate for disturbances to the system (such as changes in slope of the ground or wind speed). An ideal feedback control system cancels out all errors, effectively mitigating the effects of any forces that might or might not arise during operation and producing a response in the system that perfectly matches the user's wishes. In reality, this cannot be achieved due to measurement errors in the sensors, delays in the controller, and imperfections in the control input.

[10] I.e. a control action required for safety is not provided or is not followed, an unsafe control action is provided that leads to a hazard, a potentially safe control action is provided too late, too early, or out of sequence or a safe control action is stopped too soon (for a continuous or non-discrete control action).

5 THE ORIGINAL TOOL CHAIN AND STEP 1 OF STPA

The different tool integration aspects of the *Original Tool Chain* (OTC) are described with a Component Diagram [4], but in combination with an Activity Diagram to show the development process (see **Figure 2**).

The initial gathering and elicitation of requirements is handled by the Product Manager and the Technical Project Leader and leads up to the creation of two requirements documents, the Product Requirement Specification and the Technical Specification. These are reviewed and thereafter manually translated into four high level design documents (the System Design Specification, the Matlab System Simulation Model, the FPGA Requirements Specification and the Control Hardware Specification) by a System Architect. These are then manually translated into two detailed design documents, one for the software (the Matlab/Simulink Application Description created by a Matlab/Simulink Application Designer) and one for the hardware (the FPGA Design Specification created by a FPGA Application Engineer). A Matlab/Simulink Application Designer and a FPGA Application Engineer then collaborate to create the interface between the hardware and the software (the FPGA IO List). This interface is used during the creation of the software applications of the final product (the Control Application created by a Matlab Application Designer and the CIT Application created by a CIT Application Engineer). Parallel to this a FPGA Application Engineer develops the FPGA Application Runtime, which is applied to the actual FPGA hardware.

Most design documents (the Product Requirement Specification, the System Design Specification, etc.) from the requirements phase to the detailed design phase are written in the generic Microsoft Word tool. The only exception is the Matlab System Simulation model, which is a design model created in Matlab. In the implementation design and implementation phases the information in these design documents are transferred manually to a Matlab Simulink model, a propriety tool using IEC 61131-3 Syntax and VHDL code. Further transformations of these high-level artifacts into executable code are handled internally by some of the engineering tools (Windriver Workbench is for example used to transform the Matlab Simulink model into C Code).

All tools were deployed to individual work stations running the Microsoft Windows operating system, but with repository software running on them for supporting manual handling of versions, etc.

The details of the production of the actual FPGA hardware, the work flow of the Customer Field Engineers and most of the verification and validation activities at the integration and system levels are not included in the analyzed tool chain. This due to the details of these activities not being well enough understood, even inside the organization of our industrial partner, to allow conclusive analysis to be performed.

Two assumptions were made regarding the state of the knowledge of the staff working at our industrial partner; the first that the system architects are not experts in the dedicated design tools (such as Matlab); the second that all engineers are familiar with using the generic tools (Microsoft Word and Excel).

Table 2 and **Table 3** show the result of refining the previously defined general types of risks into the *Programmatic Risks* (see subchapter 3.1) for the OTC in the case study at hand.

6 THE SECOND ITERATION OF THE TOOL CHAIN AND STEP 1 OF STPA

The *Second Iteration of the Tool Chain* (SITC) is modeled in the same way as the OTC (see **Figure 3**).

The changes applied during this iteration were made solely to make the development process more efficient. Changes included:

- Connecting all work stations to a single intranet.
- Deploying a single repository, reachable via the intranet, for supporting manual handling of versions, etc.
- Centralizing the requirements handling to a single tool (IRQA).
- Modeling most high level and detailed design artifacts in UML using Enterprise Architect.
- Centralizing the implementation design data in a single artifact (the Matlab Target Model).
- Adding supporting transformations for interpreting high level design artifacts when creating detailed design artifacts.
- Adding transformations of detailed design artifacts via scripting to artifacts or parts of artifacts at a lower level of abstraction (for instance the creation of the FPGA Content in VHDL Code).

There were two additional assumptions made during this iteration of the tool chain; the first that all engineers are familiar with UML Modeling and UML Modeling tools; the second that the move from high level design in UML to detailed design in UML cannot be fully automated.

Table 4 and **Table 5** show the result of refining the previously defined general types of risks into the *Programmatic Risks* for the SITC in the case study at hand.

7 THE THIRD ITERATION OF THE TOOL CHAIN AND STEP 1 OF STPA

The *Third Iteration of the Tool Chain* (TITC) is modeled in the same way as the OTC (see **Figure 4**).

The changes applied during this iteration were made both to make the development process more efficient and to mitigate some of the safety issues observed in the TITC. Changes included:

- Formalizing the platform services by using an integration framework (iFEST).
- Introducing 5 life-cycle support services (a repository service, a transformation service, a traceability service, a data mining service connected to a project dashboard and a process engine).
- Introducing specialized GUIs, customized for non-expert access to specialized, engineering design tools.
- Introducing access possibilities for tool automation.
- Modeling **all** high level and detailed design artifacts in UML using Enterprise Architect.
- Updating the process to benefit from the changes (tracing between artifacts at different levels of abstraction, using the project dashboard for highlighting project problems (critical timing issues, etc.), automating transformations between process activities, etc.)

Table 6 and **Table 7** show the result of refining the previously defined general types of risks into the *Programmatic Risks* for the TITC in the case study at hand.

The next chapter continues with the second step of STPA, i.e. to determine how the unsafe control actions found in the first step can occur [2].

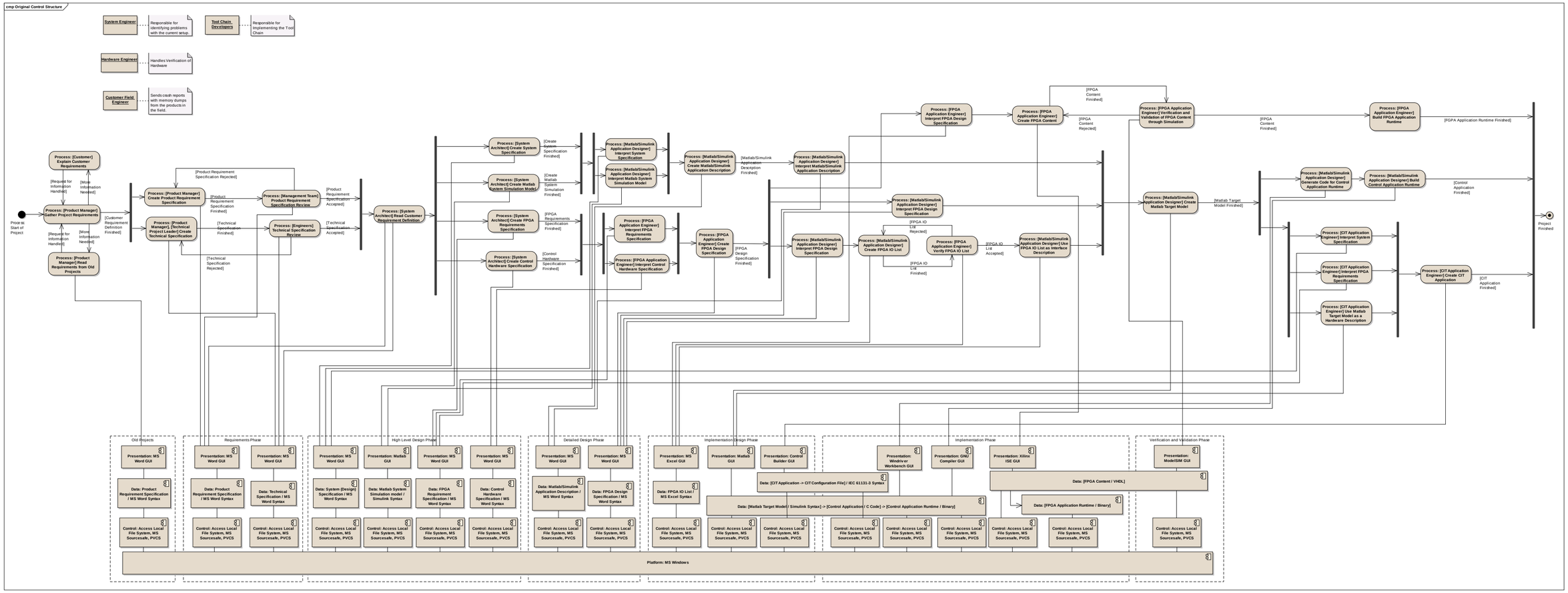

FIGURE 2 THE ORIGINAL TOOL CHAIN

	Safety-critical services cannot be provided, due to errors or missing parts in the CIS, PIS or Tool Interfaces.	Erroneous safety-critical services are provided, due to errors in the CIS, PIS or Software Tool Qualification.	Safety-critical data cannot be interpreted, due missing parts in the DIS.	Safety-critical data is corrupted, due to errors in or missing parts of the DIS, CIS, PIS or Software Tool Qualification.	Safety-critical data is not provided, due to errors in the DIS.	Safety-critical data is not possible to request / update, due to missing parts in the PIS, CIS or Tool Interfaces.	A user achieve an unintended result, due to being presented with an, to his context, unfamiliar UI.	That the state of the safety-critical data is insufficient for a specific process activity is not detected, due to erroneous or lacking possibilities to analyze the data.	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
Microsoft Windows Platform	N/A – Complete System runs on the same Platform								
Access Local File System	N/A – Platform Service on single platform								
Product Requirement Specification (PRS)	N/A	N/A	N/A – No DIS for this Artifact	(1) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(2) Customer requirements are not completely captured.	N/A
Technical Specification (TS)	N/A	N/A	N/A – No DIS for this Artifact	(3) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(4) Customer requirements are not completely captured.	N/A
System (Design) Specification (SDS)	N/A	N/A	N/A – No DIS for this Artifact	(5) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(6) PRS and/or TS not completely captured.	N/A
Matlab System Simulation Model (MSSM)	N/A	N/A	N/A – No DIS for this Artifact	(7) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	(See below) Expert UI (Matlab GUI) handled by System Architect.	(8) PRS and/or TS not completely captured.	N/A
FPGA Requirement Specification (FRS)	N/A	N/A	N/A – No DIS for this Artifact	(9) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(10) PRS and/or TS not completely captured.	N/A
Control Hardware Specification (CHS)	N/A	N/A	N/A – No DIS for this Artifact	(11) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(12) PRS and/or TS not completely captured.	N/A
Matlab/Simulink Application Description (M/SAD)	N/A	N/A	N/A – No DIS for this Artifact	(13) SDS and/or MSSM not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(14) SDS and/or MSS; model not completely captured.	N/A
FPGA Design Specification (FDS)	N/A	N/A	N/A – No DIS for this Artifact	(15) FRS and/or CHS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(16) FRS and/or CHS not completely captured.	N/A
FPGA IO List (FIOL)	N/A	N/A	N/A – No DIS for this Artifact	(17) FDS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(18) FDS not completely captured.	N/A
CIT Application to CIT Configuration File	N/A	N/A	N/A – Transformation is internal to Tool	(19) SDS, FRS and/or MTM not correctly reproduced.	N/A – Transformation is internal to Tool	N/A – Artifact only handled through UI	UI only handled by Expert	(20) SDS, FRS and/or MTM not completely captured.	N/A
Matlab Target Model (MTM) to Control Application Runtime	N/A	N/A	N/A – Transformation is internal to Tool	(21) FIOL, FDS and/or M/SAD not correctly reproduced.	N/A – Transformation is internal to Tool	N/A – Artifact only handled through UI	(See below) Expert UI (Matlab GUI) handled by CIT Application Engineer.	(22) FIOL, FDS and/or M/SAD not completely captured.	N/A

			(Windriver Workbench handles the build tool chain after configuration)		(Windriver Workbench handles the build tool chain after configuration)		(See below) Expert UI (Windriver Workbench GUI, GNU Compiler GUI) handled by Matlab/Simulink Application Designer.		
FPGA Content (FC) to FPGA Application Runtime	N/A	N/A	N/A – Transformation is internal to Tool	(23) FDS not correctly reproduced.	N/A – Transformation is internal to Tool	N/A – Artifact only handled through UI	UI only handled by Expert	(24) FDS not completely captured.	N/A
MS Word GUI	N/A – Generic UI								
Matlab GUI	(25) MSSM is created incomplete. (26) MSSM is created erroneous. (27) CIT Application is created incomplete. (28) CIT Application is created erroneous.								
MS Excel GUI	N/A – Generic UI								
Control Builder GUI	N/A – Only used by Experts								
Windriver Workbench GUI	(29) Control Application is created erroneous (erroneously configured)								
GNU Compiler GUI	(30) Control Application is created erroneous (erroneously configured)								
Xlinx ISE GUI	N/A – Only used by Experts								
ModelSIM GUI	N/A – Only used by Experts								
[Customer] Explain Customer Requirements	(31) Customer provides erroneous information and data state regress.								
Process: [Product Manager] Gather Project Requirements	(32) Requirements from Old Projects contradict New Requirements, without this being detected. (33) Product Manager decides to continue with creating the PRS and TS before the Customer Requirements are fully understood								
Process: [Product Manager] Read Requirements from Old Projects	(34) Product Manager misinterprets Requirements from Old Projects and data state regress.								
Process: [Product Manager] Create Product Requirement Specification	(35) Data to the PRS is corrupted during manual transfer.								
Process: [Product Manager], [Technical Project Leader] Create Technical Specification	(36) Data to the TS is corrupted during manual transfer.								
Process: [Management Team] Product Requirement Specification Review	(37) The state of the PRS is not sufficient, but it is accepted. (38) Management Team gives erroneous feedback and PRS state regress.								
Process: [Engineers] Technical Specification Review	(39) The state of the TS is not sufficient, but it is accepted. (40) Engineers give erroneous feedback and TS state regress.								
Process: [System Architect] Read Customer Requirement Definition	(41) The PRS and the TS contradict each other.								
Process: [System Architect] Create System Specification	(42) Data to the SDS is corrupted during manual transfer.								
Process: [System Architect] Create Matlab System Simulation Model	(43) Data to the MSSM is corrupted during manual transfer.								
Process: [System Architect] Create FPGA Requirements Specification	(44) Data to the FRS is corrupted during manual transfer.								
Process: [System Architect] Create Control Hardware Specification	(45) Data to the CHS is corrupted during manual transfer.								
Process: [Matlab/Simulink Application Designer] Interpret System Specification	(46) The state of the SDS is not sufficient, but it is used.								
Process: [Matlab/Simulink Application	(47) The state of the MSSM is not sufficient, but it is used.								

Designer] Interpret Matlab System Simulation Model	
Process: [FPGA Application Engineer] Interpret FPGA Requirements Specification	(48) The state of the FRS is not sufficient, but it is used.
Process: [FPGA Application Engineer] Interpret Control Hardware Specification	(49) The state of the CHS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Create Matlab/Simulink Application Description	(50) The SDS and the MSSM contradict each other. (51) Data to the M/SAD is corrupted during manual transfer.
Process: [FPGA Application Engineer] Create FPGA Design Specification	(52) The FRS and the CHS contradict each other. (53) Data to the FDS is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Interpret Matlab/Simulink Application Description	(54) The state of the M/SAD is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design Specification	(55) The state of the FDS is not sufficient, but it is used.
Process: [FPGA Application Engineer] Interpret FPGA Design Specification	(56) The state of the FDS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design Specification	(57) The state of the FDS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Create FPGA IO List	(58) Data to the FIOL is corrupted during manual transfer.
Process: [FPGA Application Engineer] Verify FPGA IO List	(59) The state of the FIOL is not sufficient, but it is accepted. (60) FPGA Application Engineer gives erroneous feedback and FIOL state regress.
Process: [FPGA Application Engineer] Create FPGA Content	(61) Data to the FC is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Use FPGA IO List as Interface Description	(62) The state of the FIOL is not sufficient, but it is used.
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	(63) The state of the FC is not sufficient, but it is accepted.
Process: [FPGA Application Engineer] Build FPGA Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab Target Model	(64) The FDS, the M/SAD and the FIOL contradict each other. (65) The FDS is interpreted differently in different process branches. (66) Data to the MTM is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Generate Code for Control Application Runtime	(67) The state of the MTM is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Build Control Application	N/A – No probable risk in regard to our context

Runtime	
Process: [CIT Application Engineer] Interpret System Specification	(68) The state of the SDS is not sufficient, but it is used.
Process: [CIT Application Engineer] Interpret FPGA Requirements Specification	(69) The state of the FRS is not sufficient, but it is used.
Process: [CIT Application Engineer] Use as Matlab Target Model as a Hardware Description	(70) The state of the MTM is not sufficient, but it is used.
Process: [CIT Application Engineer] Create CIT Application (CA)	(71) The SDS, the FRS and the MTM contradict each other. (72) The SDS is interpreted differently in different process branches. (73) The FRS is interpreted differently in different process branches. (74) Data to the CA is corrupted during manual transfer.

TABLE 2 ORIGINAL TOOL CHAIN, PROGRAMMATIC RISKS, PART 1

	Safety-critical services are invoked too early or too late, due to timing issues in or missing parts of the CIS or PIS.	Safety-critical services are stopped too soon or applied too long, due to timing issues in or missing parts of the CIS or PIS.	Safety-critical data is not complete when provided, due to timing issues in or missing parts of the CIS, PIS or DIS.	Safety-critical data is provided too early or too late, due to timing issues in or missing parts of the CIS, PIS or DIS.	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
[Customer] Explain Customer Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Gather Project Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Read Requirements from Old Projects	N/A – No probable risk in regard to our context				
Process: [Product Manager] Create Product Requirement Specification	N/A – No probable risk in regard to our context				
Process: [Product Manager], [Technical Project Leader] Create Technical Specification	N/A – No probable risk in regard to our context				
Process: [Management Team] Product Requirement Specification Review	N/A – No probable risk in regard to our context				
Process: [Engineers] Technical Specification Review	N/A – No probable risk in regard to our context				
Process: [System Architect] Read Customer Requirement Definition	N/A – No probable risk in regard to our context				
Process: [System Architect] Create System Specification	(75) To create the SDS takes too long time, due to manual transformation of data. An incomplete version of the SDS is therefore used.				
Process: [System Architect] Create Matlab System Simulation Model	(76) To create the MSSM takes too long time, due to manual transformation of data. An incomplete version of the MSSM is therefore used.				
Process: [System Architect] Create FPGA Requirements Specification	(77) To create the FRS takes too long time, due to manual transformation of data. An incomplete version of the FRS is therefore used.				

Process: [System Architect] Create Control Hardware Specification	(78) To create the CHS takes too long time, due to manual transformation of data. An incomplete version of the CHS is therefore used.
Process: [Matlab/Simulink Application Designer] Interpret System Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret Matlab System Simulation Model	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret Control Hardware Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab/Simulink Application Description	(79) To create the M/SAD takes too long time, due to manual transformation of data. An incomplete version of the M/SAD is therefore used.
Process: [FPGA Application Engineer] Create FPGA Design Specification	(80) To create the FDS takes too long time, due to manual transformation of data. An incomplete version of the FDS is therefore used.
Process: [Matlab/Simulink Application Designer] Interpret Matlab/Simulink Application Description	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret FPGA Design Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create FPGA IO List	(81) To create the FIOL takes too long time, due to manual transformation of data. An incomplete version of the FIOL is therefore used.
Process: [FPGA Application Engineer] Verify FPGA IO List	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Create FPGA Content	(82) To create the FC takes too long time, due to manual transformation of data. An incomplete version of the FC is therefore used.
Process: [Matlab/Simulink Application Designer] Use FPGA IO List as Interface Description	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	(83) Simulation ended too soon, due to manual decision. A suboptimal/erroneous version of the FC is therefore used.
Process: [FPGA Application Engineer] Build FPGA Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab Target Model	(84) To create the MTM takes too long time, due to manual transformation of data. An incomplete version of the MTM is therefore used.

Process: [Matlab/Simulink Application Designer] Generate Code for Control Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Build Control Application Runtime	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret System Specification	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Use as Matlab Target Model as a Hardware Description	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Create CIT Application	(85) To create the CA takes too long time, due to manual transformation of data. An incomplete version of the CA is therefore used.

TABLE 3 ORIGINAL TOOL CHAIN, PROGRAMMATIC RISKS, PART 2

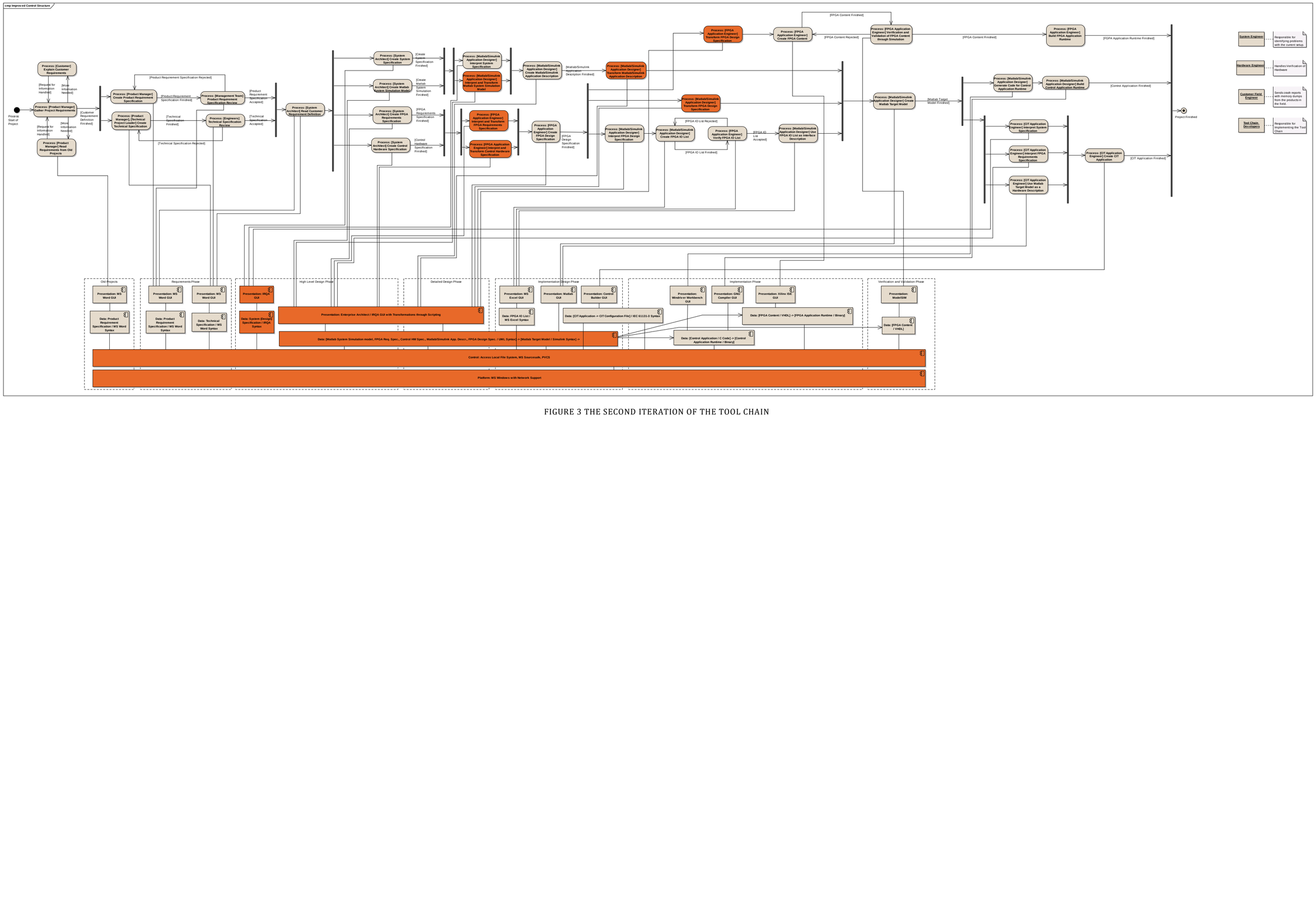

FIGURE 3 THE SECOND ITERATION OF THE TOOL CHAIN

	Safety-critical services cannot be provided, due to errors or missing parts in the CIS, PIS or Tool Interfaces.	Erroneous safety-critical services are provided, due to errors in the CIS, PIS or Software Tool Qualification.	Safety-critical data cannot be interpreted, due missing parts in the DIS.	Safety-critical data is corrupted, due to errors in or missing parts of the DIS, CIS, PIS or Software Tool Qualification.	Safety-critical data is not provided, due to errors in the DIS.	Safety-critical data is not possible to request / update, due to missing parts in the PIS, CIS or Tool Interfaces.	A user achieve an unintended result, due to being presented with an, to his context, unfamiliar UI.	That the state of the safety-critical data is insufficient for a specific process activity is not detected, due to erroneous or lacking possibilities to analyze the data.	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
Microsoft Windows Platform	N/A – Complete System runs on the same Platform								
Access Local File System	N/A – Platform Service on single platform								
Product Requirement Specification (PRS)	N/A	N/A	N/A – No DIS for this Artifact	(1) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(2) Customer requirements are not completely captured.	N/A
Technical Specification (TS)	N/A	N/A	N/A – No DIS for this Artifact	(3) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(4) Customer requirements are not completely captured.	N/A
System (Design) Specification (SDS)	N/A	N/A	N/A – No DIS for this Artifact	(5) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	(See below) Expert UI (IRQA GUI) handled by Matlab/Simulink Application Designer. (See below) Expert UI (IRQA GUI) handled by CIT Application Engineer.	(6) PRS and/or TS not completely captured.	N/A
Matlab System Simulation Model (MSSM)	N/A	N/A	N/A – No DIS for this Artifact	(7) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	(See below) Expert UI (IRQA GUI) handled by Matlab/Simulink Application Designer.	(8) PRS and/or TS not completely captured.	N/A
FPGA Requirement Specification (FRS)	N/A	N/A	N/A – No DIS for this Artifact	(9) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	(See below) Expert UI (IRQA GUI) handled by FPGA Application Engineer. (See below) Expert UI (IRQA GUI) handled by CIT Application Engineer.	(10) PRS and/or TS not completely captured.	N/A
Control Hardware Specification (CHS)	N/A	N/A	N/A – No DIS for this Artifact	(11) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	(See below) Expert UI (IRQA GUI) handled by FPGA Application Engineer.	(12) PRS and/or TS not completely captured.	N/A
Matlab/Simulink Application Description (M/SAD)	N/A	N/A	(86) Required transformations to M/SAD not constructed prior to launch of tool chain.	(13) SDS not correctly reproduced, (87) The M/SAD is created erroneous		N/A – Artifact only handled through UI	UI only handled by Experts	(14) SDS and/or MSSM model not completely captured.	N/A

FPGA Design Specification (FDS)	N/A	N/A	(88) Required transformations to FDS not constructed prior to launch of tool chain.	(89) The FDS is created erroneous		N/A – Artifact only handled through UI	UI only handled by Experts	(16) FRS and/or CHS not completely captured.	N/A
FPGA IO List (FIOL)	N/A	N/A	N/A – No DIS for this Artifact	(17) FDS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(18) FDS not completely captured.	N/A
CIT Application to CIT Configuration File	N/A	N/A	N/A – Transformation is internal to Tool	(19) SDS, FRS and/or MTM not correctly reproduced.	N/A – Transformation is internal to Tool	N/A – Artifact only handled through UI	UI only handled by Expert	(20) SDS, FRS and/or MTM not completely captured.	N/A
Matlab Target Model (MTM) to Control Application to Control Application Runtime	N/A	N/A	(90) Required transformations to MTM not constructed prior to launch of tool chain. (91) Required transformations to Control Application not constructed prior to launch of tool chain.	(21) FIOL not correctly reproduced, (92) The MTM is created erroneous, (93) The Control Application is created erroneous		N/A – Artifact only handled through UI	(See below) Expert UI (Matlab GUI) handled by CIT Application Engineer. (See below) Expert UI (Windriver Workbench GUI, GNU Compiler GUI) handled by Matlab/Simulink Application Designer.	(22) FIOL not completely captured.	N/A
FPGA Design Specification (FDS) to FPGA Content (FC) to FPGA Application Runtime	N/A	N/A	(94) Required transformations to FC not constructed prior to launch of tool chain.	(95) The FC is created erroneous		N/A – Artifact only handled through UI	UI only handled by Expert	Transformations of FDS.	N/A
MS Word GUI	N/A – Generic UI								
IRQA GUI	(96) M/SAD is created incomplete. (97) M/SAD is created erroneous. (98) CIT Application is created incomplete. (99) CIT Application is created erroneous. (100) FDS is created incomplete. (101) FDS is created erroneous.								
Enterprise Architect GUI	N/A – Only used by Experts								
Matlab GUI	(27) CIT Application is created incomplete. (28) CIT Application is created erroneous.								
MS Excel GUI	N/A – Generic UI								
Control Builder GUI	N/A – Only used by Experts								
Windriver Workbench GUI	(29) Control Application is created erroneous (erroneously configured)								
GNU Compiler GUI	(30) Control Application is created erroneous (erroneously configured)								
Xlinx ISE GUI	N/A – Only used by Experts								
ModelSIM GUI	N/A – Only used by Experts								
[Customer] Explain Customer Requirements	(31) Customer provides erroneous information and data state regress.								
Process: [Product Manager] Gather Project Requirements	(32) Requirements from Old Projects contradict New Requirements, without this being detected. (33) Product Manager decides to continue with creating the PRS and TS before the Customer Requirements are fully understood								
Process: [Product Manager] Read Requirements from Old Projects	(34) Product Manager misinterprets Requirements from Old Projects and data state regress.								
Process: [Product Manager] Create Product Requirement Specification	(35) Data to the PRS is corrupted during manual transfer.								
Process: [Product Manager], [Technical Project Leader] Create Technical Specification	(36) Data to the TS is corrupted during manual transfer.								

Process: [Management Team] Product Requirement Specification Review	(37) The state of the PRS is not sufficient, but it is accepted. (38) Management Team gives erroneous feedback and PRS state regress.
Process: [Engineers] Technical Specification Review	(39) The state of the TS is not sufficient, but it is accepted. (40) Engineers give erroneous feedback and TS state regress.
Process: [System Architect] Read Customer Requirement Definition	(41) The PRS and the TS contradict each other.
Process: [System Architect] Create System Specification	(42) Data to the SDS is corrupted during manual transfer.
Process: [System Architect] Create Matlab System Simulation Model	(43) Data to the MSSM is corrupted during manual transfer.
Process: [System Architect] Create FPGA Requirements Specification	(44) Data to the FRS is corrupted during manual transfer.
Process: [System Architect] Create Control Hardware Specification	(45) Data to the CHS is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Interpret System Specification	(46) The state of the SDS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Interpret and Transform Matlab System Simulation Model	(47) The state of the MSSM is not sufficient, but it is used.
Process: [FPGA Application Engineer] Interpret and Transform FPGA Requirements Specification	(48) The state of the FRS is not sufficient, but it is used.
Process: [FPGA Application Engineer] Interpret and Transform Control Hardware Specification	(49) The state of the CHS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Create Matlab/Simulink Application Description	(50) The SDS and the MSSM contradict each other. (51) Data to the M/SAD is corrupted during manual transfer.
Process: [FPGA Application Engineer] Create FPGA Design Specification	(52) The FRS and the CHS contradict each other. (53) Data to the FDS is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Transform Matlab/Simulink Application Description	(54) The state of the M/SAD is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Transform FPGA Design Specification	(55) The state of the FDS is not sufficient, but it is used.
Process: [FPGA Application Engineer] Transform FPGA Design Specification	(56) The state of the FDS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design Specification	(57) The state of the FDS is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Create FPGA IO List	(58) Data to the FIOL is corrupted during manual transfer.
Process: [FPGA Application Engineer]	(59) The state of the FIOL is not sufficient, but it is accepted. (60) FPGA Application Engineer gives erroneous feedback and FIOL state regress.

Verify FPGA IO List	
Process: [FPGA Application Engineer] Create FPGA Content	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Use FPGA IO List as Interface Description	(62) The state of the FIOL is not sufficient, but it is used.
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	(63) The state of the FC is not sufficient, but it is accepted.
Process: [FPGA Application Engineer] Build FPGA Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab Target Model	(64) The FDS, the M/SAD and the FIOL contradict each other. (65) The FDS is interpreted differently in different process branches. (66) Data to the MTM is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Generate Code for Control Application Runtime	(67) The state of the MTM is not sufficient, but it is used.
Process: [Matlab/Simulink Application Designer] Build Control Application Runtime	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret System Specification	(68) The state of the SDS is not sufficient, but it is used.
Process: [CIT Application Engineer] Interpret FPGA Requirements Specification	(69) The state of the FRS is not sufficient, but it is used.
Process: [CIT Application Engineer] Use as Matlab Target Model as a Hardware Description	(70) The state of the MTM is not sufficient, but it is used.
Process: [CIT Application Engineer] Create CIT Application (CA)	(71) The SDS, the FRS and the MTM contradict each other. (72) The SDS is interpreted differently in different process branches. (73) The FRS is interpreted differently in different process branches. (74) Data to the CA is corrupted during manual transfer.

TABLE 4 THE SECOND ITERATION OF THE TOOL CHAIN, PROGRAMMATIC RISKS, PART 1

	Safety-critical services are invoked too early or too late, due to timing issues in or missing parts of the CIS or PIS.	Safety-critical services are stopped too soon or applied too long, due to timing issues in or missing parts of the CIS or PIS.	Safety-critical data is not complete when provided, due to timing issues in or missing parts of the CIS, PIS or DIS.	Safety-critical data is provided too early or too late, due to timing issues in or missing parts of the CIS, PIS or DIS.	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
[Customer] Explain Customer Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Gather Project Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Read Requirements from Old Projects	N/A – No probable risk in regard to our context				
Process: [Product Manager] Create	N/A – No probable risk in regard to our context				

Product Requirement Specification	
Process: [Product Manager], [Technical Project Leader] Create Technical Specification	N/A – No probable risk in regard to our context
Process: [Management Team] Product Requirement Specification Review	N/A – No probable risk in regard to our context
Process: [Engineers] Technical Specification Review	N/A – No probable risk in regard to our context
Process: [System Architect] Read Customer Requirement Definition	N/A – No probable risk in regard to our context
Process: [System Architect] Create System Specification	(75) To create the SDS takes too long time, due to manual transformation of data. An incomplete version of the SDS is therefore used.
Process: [System Architect] Create Matlab System Simulation Model	(76) To create the MSSM takes too long time, due to manual transformation of data. An incomplete version of the MSSM is therefore used.
Process: [System Architect] Create FPGA Requirements Specification	(77) To create the FRS takes too long time, due to manual transformation of data. An incomplete version of the FRS is therefore used.
Process: [System Architect] Create Control Hardware Specification	(78) To create the CHS takes too long time, due to manual transformation of data. An incomplete version of the CHS is therefore used.
Process: [Matlab/Simulink Application Designer] Interpret System Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret and Transform Matlab System Simulation Model	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret and Transform FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret and Transform Control Hardware Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab/Simulink Application Description	(79) To create the M/SAD takes too long time, due to manual transformation of data. An incomplete version of the M/SAD is therefore used. (102) To create the M/SAD takes too long time, due to the data integration transformations. An incomplete version of the M/SAD is therefore used.
Process: [FPGA Application Engineer] Create FPGA Design Specification	(80) To create the FDS takes too long time, due to manual transformation of data. An incomplete version of the FDS is therefore used. (103) To create the FDS takes too long time, due to the data integration transformations. An incomplete version of the FDS is therefore used.
Process: [Matlab/Simulink Application Designer] Transform Matlab/Simulink Application Description	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Transform FPGA Design Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Transform FPGA Design Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret FPGA Design	N/A – No probable risk in regard to our context

Specification	
Process: [Matlab/Simulink Application Designer] Create FPGA IO List	(81) To create the FIOL takes too long time, due to manual transformation of data. An incomplete version of the FIOL is therefore used.
Process: [FPGA Application Engineer] Verify FPGA IO List	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Create FPGA Content	(104) To create the FC takes too long time, due to the data integration transformations. An incomplete version of the FC is therefore used.
Process: [Matlab/Simulink Application Designer] Use FPGA IO List as Interface Description	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	(83) Simulation ended too soon, due to manual decision. A suboptimal/erroneous version of the FC is therefore used.
Process: [FPGA Application Engineer] Build FPGA Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create Matlab Target Model	(84) To create the MTM takes too long time, due to manual transformation of data. An incomplete version of the MTM is therefore used. (105) To create the MTM takes too long time, due to the data integration transformations. An incomplete version of the MTM is therefore used.
Process: [Matlab/Simulink Application Designer] Generate Code for Control Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Build Control Application Runtime	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret System Specification	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Use as Matlab Target Model as a Hardware Description	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Create CIT Application (CA)	(85) To create the CA takes too long time, due to manual transformation of data. An incomplete version of the CA is therefore used.

TABLE 5 THE SECOND ITERATION OF THE TOOL CHAIN, PROGRAMMATIC RISKS, PART 2

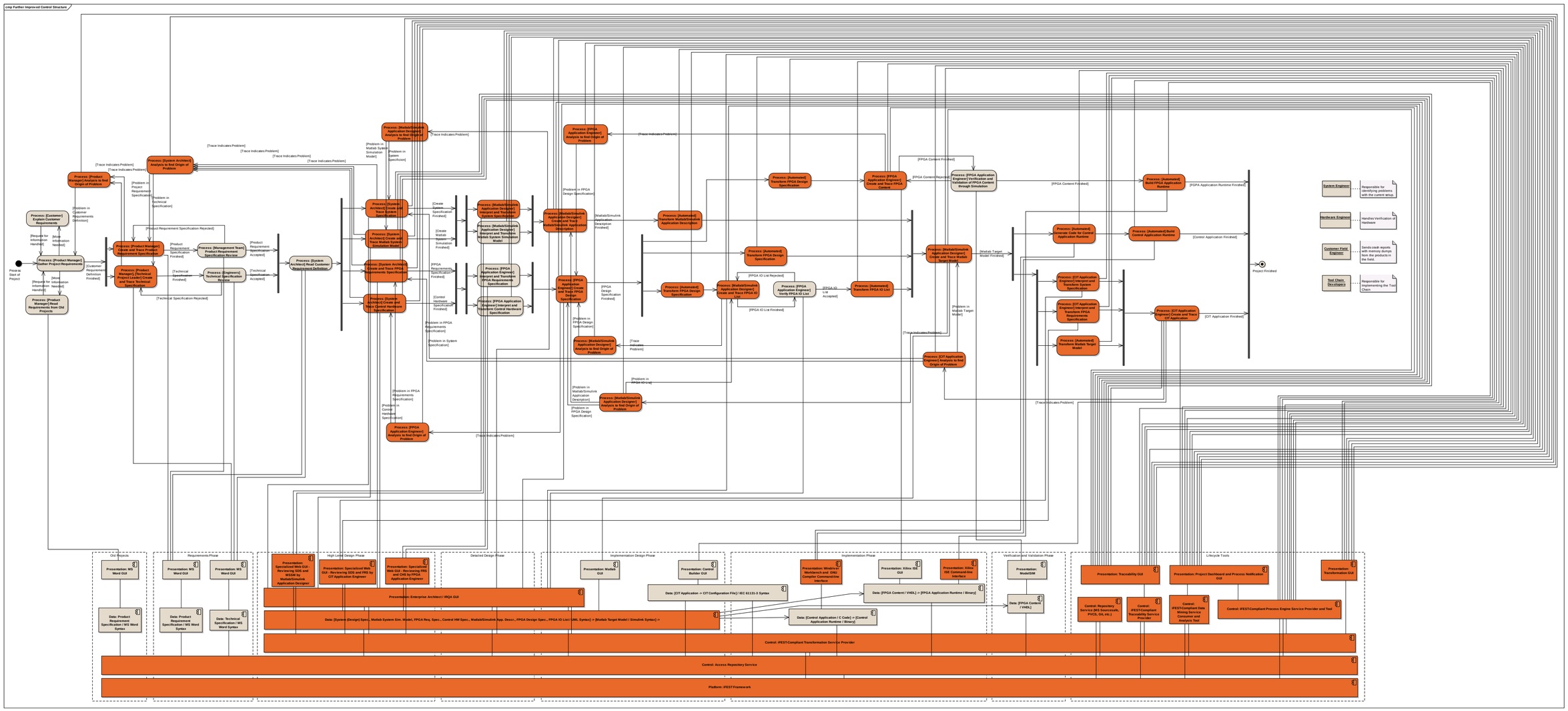

FIGURE 4 THE THIRD ITERATION OF THE TOOL CHAIN

	Safety-critical services cannot be provided, due to errors or missing parts in the CIS, PIS or Tool Interfaces.	Erroneous safety-critical services are provided, due to errors in the CIS, PIS or Software Tool Qualification.	Safety-critical data cannot be interpreted, due missing parts in the DIS.	Safety-critical data is corrupted, due to errors in or missing parts of the DIS, CIS, PIS or Software Tool Qualification.	Safety-critical data is not provided, due to errors in the DIS.	Safety-critical data is not possible to request / update, due to missing parts in the PIS, CIS or Tool Interfaces.	A user achieve an unintended result, due to being presented with an, to his context, unfamiliar UI.	That the state of the safety-critical data is insufficient for a specific process activity is not detected, due to erroneous or lacking possibilities to analyze the data.	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
iFEST Framework Platform	N/A – Complete System runs on the same Platform								
Access Repository Service	N/A – Platform Service on single platform								
Product Requirement Specification (PRS)	N/A	N/A	N/A – No DIS for this Artifact	(1) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(2) Customer requirements are not completely captured.	N/A
Technical Specification (TS)	N/A	N/A	N/A – No DIS for this Artifact	(3) Customer requirements are not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Generic UI	(4) Customer requirements are not completely captured.	N/A
System (Design) Specification (SDS)	N/A	N/A	N/A – No DIS for this Artifact	(5) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Access only by Experts or through Specialized UI	(6) PRS and/or TS not completely captured.	N/A
Matlab System Simulation Model (MSSM)	N/A	N/A	N/A – No DIS for this Artifact	(7) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Access only by Experts or through Specialized UI	(8) PRS and/or TS not completely captured.	N/A
FPGA Requirement Specification (FRS)	N/A	N/A	N/A – No DIS for this Artifact	(9) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Access only by Experts or through Specialized UI	(10) PRS and/or TS not completely captured.	N/A
Control Hardware Specification (CHS)	N/A	N/A	N/A – No DIS for this Artifact	(11) PRS and/or TS not correctly reproduced.	N/A – No DIS for this Artifact	N/A – Artifact only handled through UI	Access only by Experts or through Specialized UI	(12) PRS and/or TS not completely captured.	N/A
Matlab/Simulink Application Description (M/SAD)	N/A	N/A	N/A - Issue centralized by Transformation Service	N/A – Issues are centralized by Transformation and Repository Services			UI only handled by Experts	(14) SDS and/or MSSM model not completely captured.	N/A
FPGA Design Specification (FDS)	N/A	N/A	N/A - Issue centralized by Transformation Service	N/A – Issues are centralized by Transformation and Repository Services			UI only handled by Experts	(16) FRS and/or CHS not completely captured.	N/A
FPGA IO List (FIOL)	N/A	N/A	N/A - Issue centralized by Transformation Service	N/A – Issues are centralized by Transformation and Repository Services			Generic UI	Transformations of FDS.	N/A
CIT Application to CIT Configuration File	N/A	N/A	N/A – Transformation is internal to Tool	(19) SDS and/or FRS not correctly reproduced.	N/A – Transformation is internal to Tool	N/A – Artifact only handled through UI	UI only handled by Expert	(20) SDS and/or FRS not completely captured.	N/A
Matlab Target Model (MTM) to Control Application to Control Application Runtime	N/A	N/A	N/A - Issue centralized by Transformation Service	N/A – Issues are centralized by Transformation and Repository Services			Access only by Experts, through Specialized UI or by Automated Tools.	Transformations of FIOL.	N/A
Matlab Target Model (MTM) to FPGA Content (FC) to FPGA Application Runtime	N/A	N/A	N/A - Issue centralized by Transformation Service.	N/A – Issues are centralized by Transformation and Repository Services		N/A – Artifact only handled through UI (and the build process is not possible to bypass manually)	UI only handled by Expert	Transformations of FDS.	N/A
iFEST-Compliant Transformation Service	(106) Required	(107) Parts of	(108) Required	(109) Artifact is created erroneous		(110) Artifact not	N/A	N/A	N/A

Provider	control not constructed prior to launch of tool chain	services lose safety certification, but are not removed from service catalogue	transformation not constructed prior to launch of tool chain		correctly reproduced.			
Repository Service (MS Sourcesafe, PVCS, Git, etc.)	(111) Required control not constructed prior to launch of tool chain	(112) Parts of services lose safety certification, but are not removed from service catalogue	N/A	(113) Artifact is created erroneous	(114) Artifact not correctly reproduced.	N/A	N/A	N/A
iFEST-Compliant Traceability Service Provider	(115) Required control not constructed prior to launch of tool chain	(116) Parts of services lose safety certification, but are not removed from service catalogue	N/A	(117) Artifact is left in erroneous state (incorrect feedback)		N/A	N/A	N/A
iFEST-Compliant Data Mining Service Consumer and Analysis Tool	(118) Required control not constructed prior to launch of tool chain	(119) Parts of services lose safety certification, but are not removed from service catalogue	(120) Artifact is left in erroneous state (incorrect feedback)			N/A	(121) Artifact is left in erroneous state (incorrect feedback)	N/A
iFEST-Compliant Process Engine Service Provider and Tool	(122) Required control not constructed prior to launch of tool chain	(123) Parts of services lose safety certification, but are not removed from service catalogue	N/A			N/A	N/A	(124) Required process logic not constructed prior to launch of tool chain
MS Word GUI	N/A – Generic UI							
IRQA GUI	N/A - Access only by Experts							
Enterprise Architect GUI	N/A - Access only by Experts							
Specialized Web GUI - Reviewing SDS and MSSM by Matlab/Simulink Application Designer	N/A - Access through Specialized UI							
Specialized Web GUI - Reviewing SDS and FRS by CIT Application Engineer	N/A - Access through Specialized UI							
Specialized Web GUI - Reviewing FRS and CHS by FPGA Application Engineer	N/A - Access through Specialized UI							
Matlab GUI	N/A - Access only by Experts							
MS Excel GUI	N/A – Generic UI							
Control Builder GUI	N/A – Only used by Experts							
Windriver Workbench and GNU Compiler Command-line Interface	N/A – Only Automated Access							
Xlinx ISE GUI	N/A – Only used by Experts							
Xilinx ISE Command-line Interface	N/A – Only Automated Access							
ModelSIM GUI	N/A – Only used by Experts							
Traceability GUI	N/A – Generic UI							
Project Dashboard and Process Notification GUI	N/A – Generic UI							

Transformation GUI	N/A – Generic UI
[Customer] Explain Customer Requirements	(31) Customer provides erroneous information and data state regress.
Process: [Product Manager] Gather Project Requirements	(32) Requirements from Old Projects contradict New Requirements, without this being detected. (33) Product Manager decides to continue with creating the PRS and TS before the Customer Requirements are fully understood
Process: [Product Manager] Read Requirements from Old Projects	(34) Product Manager misinterprets Requirements from Old Projects and data state regress.
Process: [Product Manager] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Product Manager] Create and Trace Product Requirement Specification	(35) Data to the PRS is corrupted during manual transfer.
Process: [Product Manager], [Technical Project Leader] Create and Trace Technical Specification	(36) Data to the TS is corrupted during manual transfer.
Process: [System Architect] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Management Team] Product Requirement Specification Review	(37) The state of the PRS is not sufficient, but it is accepted. (38) Management Team gives erroneous feedback and PRS state regress.
Process: [Engineers] Technical Specification Review	(39) The state of the TS is not sufficient, but it is accepted. (40) Engineers give erroneous feedback and TS state regress.
Process: [System Architect] Read Customer Requirement Definition	(41) The PRS and the TS contradict each other.
Process: [System Architect] Create and Trace System Specification	(42) Data to the SDS is corrupted during manual transfer.
Process: [System Architect] Create and Trace Matlab System Simulation Model	(43) Data to the MSSM is corrupted during manual transfer.
Process: [System Architect] Create and Trace FPGA Requirements Specification	(44) Data to the FRS is corrupted during manual transfer.
Process: [System Architect] Create and Trace Control Hardware Specification	(45) Data to the CHS is corrupted during manual transfer.
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [FPGA Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Matlab/Simulink Application Designer] Interpret and Transform System Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Interpret and Transform Matlab System Simulation Model	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret and Transform FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret and Transform Control Hardware Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create and Trace Matlab/Simulink Application Description	(51) Data to the M/SAD is corrupted during manual transfer.
Process: [FPGA Application Engineer] Create and Trace FPGA Design Specification	(53) Data to the FDS is corrupted during manual transfer.

Process: [FPGA Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Automated] Transform Matlab/Simulink Application Description (Top Middle)	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [Automated] Transform FPGA Design Specification (Bottom Middle)	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [Automated] Transform FPGA Design Specification (Top)	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [Automated] Transform FPGA Design Specification (Bottom)	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [Matlab/Simulink Application Designer] Create and Trace FPGA IO List	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Verify FPGA IO List	(60) FPGA Application Engineer gives erroneous feedback and FIOL state regress.
Process: [FPGA Application Engineer] Create and Trace FPGA Content	N/A – No probable risk in regard to our context
Process: [Automated] Transform FPGA IO List	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	N/A – No probable risk in regard to our context
Process: [Automated] Build FPGA Application Runtime	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create and Trace Matlab Target Model	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((120) Artifact is left in erroneous state (incorrect feedback), etc.)
Process: [Automated] Generate Code for Control Application Runtime	N/A – No probable risk in regard to our context
Process: [Automated] Build Control Application Runtime	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret and Transform System Specification	N/A – No probable risk in regard to our context
Process: [CIT Application Engineer] Interpret and Transform FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [Automated] Transform Matlab Target Model	N/A – Centralized Risk ((109) Artifact is created erroneous, etc.)
Process: [CIT Application Engineer] Create and Trace CIT Application	(74) Data to the CA is corrupted during manual transfer.

TABLE 6 THE THIRD ITERATION OF THE TOOL CHAIN, PROGRAMMATIC RISKS, PART 1

	Safety-critical services are invoked too early or too	Safety-critical services are stopped too soon or applied too	Safety-critical data is not complete when provided,	Safety-critical data is provided too early or too late, due to	Safety-critical services are erroneously invoked, fail to be invoked, invoked in the

	late, due to timing issues in or missing parts of the CIS or PIS.	long, due to timing issues in or missing parts of the CIS or PIS.	due to timing issues in or missing parts of the CIS, PIS or DIS.	timing issues in or missing parts of the CIS, PIS or DIS.	wrong sequence or applied too long / stopped too soon, due to erroneous CIS or erroneous, misunderstood or missing process logic.
iFEST-Compliant Transformation Service Provider	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
Repository Service (MS Sourcesafe, PVCS, Git, etc.)	(125) Artifact is versioned erroneously and versions are mistaken for each other				
iFEST-Compliant Traceability Service Provider	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
iFEST-Compliant Data Mining Service Consumer and Analysis Tool	(126) To provide feedback takes too long time, due to timing issues with the data mining and necessary feedback is not used				
iFEST-Compliant Process Engine Service Provider and Tool	(127) Notifications are out-of-sync with actual events and wrong version of artifacts are used, (128) Notifications are out-of-sync with actual events and artifacts are not updated accordingly, (129) Notifications are not delivered correctly and necessary feedback is not used				
[Customer] Explain Customer Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Gather Project Requirements	N/A – No probable risk in regard to our context				
Process: [Product Manager] Read Requirements from Old Projects	N/A – No probable risk in regard to our context				
Process: [Product Manager] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
Process: [Product Manager] Create and Trace Product Requirement Specification	N/A – Centralized Risk ((125) Notifications are not delivered correctly and necessary feedback is not used, etc.)				
Process: [Product Manager], [Technical Project Leader] Create and Trace Technical Specification	N/A – Centralized Risk ((125) Notifications are not delivered correctly and necessary feedback is not used, etc.)				
Process: [System Architect] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
Process: [Management Team] Product Requirement Specification Review	N/A – No probable risk in regard to our context				
Process: [Engineers] Technical Specification Review	N/A – No probable risk in regard to our context				
Process: [System Architect] Read Customer Requirement Definition	N/A – No probable risk in regard to our context				
Process: [System Architect] Create and Trace System Specification	(75) To create the SDS takes too long time, due to manual transformation of data. An incomplete version of the SDS is therefore used.				
Process: [System Architect] Create and Trace Matlab System Simulation Model	(76) To create the MSSM takes too long time, due to manual transformation of data. An incomplete version of the MSSM is therefore used.				
Process: [System Architect] Create and Trace FPGA Requirements Specification	(77) To create the FRS takes too long time, due to manual transformation of data. An incomplete version of the FRS is therefore used.				
Process: [System Architect] Create and Trace Control Hardware Specification	(78) To create the CHS takes too long time, due to manual transformation of data. An incomplete version of the CHS is therefore used.				
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
Process: [FPGA Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)				
Process: [Matlab/Simulink Application Designer] Interpret and Transform System Specification	N/A – No probable risk in regard to our context				
Process: [Matlab/Simulink Application Designer] Interpret and Transform Matlab	N/A – No probable risk in regard to our context				

System Simulation Model	
Process: [FPGA Application Engineer] Interpret and Transform FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [FPGA Application Engineer] Interpret and Transform Control Hardware Specification	N/A – No probable risk in regard to our context
Process: [Matlab/Simulink Application Designer] Create and Trace Matlab/Simulink Application Description	(79) To create the M/SAD takes too long time, due to manual transformation of data. An incomplete version of the M/SAD is therefore used. (102) To create the M/SAD takes too long time, due to the data integration transformations. An incomplete version of the M/SAD is therefore used.
Process: [FPGA Application Engineer] Create and Trace FPGA Design Specification	(80) To create the FDS takes too long time, due to manual transformation of data. An incomplete version of the FDS is therefore used. (103) To create the FDS takes too long time, due to the data integration transformations. An incomplete version of the FDS is therefore used.
Process: [FPGA Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Matlab/Simulink Application Designer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Transform Matlab/Simulink Application Description	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Transform FPGA Design Specification	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Transform FPGA Design Specification	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Transform FPGA Design Specification	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Transform FPGA Design Specification	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Matlab/Simulink Application Designer] Create and Trace FPGA IO List	(130) To create the FIOL takes too long time, due to the data integration transformations. An incomplete version of the FIOL is therefore used.
Process: [FPGA Application Engineer] Verify FPGA IO List	N/A – Centralized Risk ((129) Notifications are not delivered correctly and necessary feedback is not used, etc.)
Process: [FPGA Application Engineer] Create and Trace FPGA Content	(104) To create the FC takes too long time, due to the data integration transformations. An incomplete version of the FC is therefore used.
Process: [Automated] Transform FPGA IO List	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [FPGA Application Engineer] Verification and Validation of FPGA Content through Simulation	(83) Simulation ended too soon, due to manual decision. A suboptimal/erroneous version of the FC is therefore used.
Process: [Automated] Build FPGA Application Runtime	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Matlab/Simulink Application Designer] Create and Trace Matlab Target Model	(105) To create the MTM takes too long time, due to the data integration transformations. An incomplete version of the MTM is therefore used.
Process: [CIT Application Engineer] Analysis to find Origin of Problem	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Generate Code for Control Application Runtime	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [Automated] Build Control Application Runtime	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [CIT Application Engineer] Interpret and Transform System	N/A – No probable risk in regard to our context

Specification	
Process: [CIT Application Engineer] Interpret and Transform FPGA Requirements Specification	N/A – No probable risk in regard to our context
Process: [Automated] Transform Matlab Target Model	N/A – Centralized Risk ((125) Artifact is versioned erroneously and versions are mistaken for each other, etc.)
Process: [CIT Application Engineer] Create and Trace CIT Application	(85) To create the CA takes too long time, due to manual transformation of data. An incomplete version of the CA is therefore used. (131) To create the CA takes too long time, due to the data integration transformations. An incomplete version of the CA is therefore used.

TABLE 7 THE THIRD ITERATION OF THE TOOL CHAIN, PROGRAMMATIC RISKS, PART 2

8 STEP 2 OF STPA

The first part of the second step of a STPA is to determine how the unsafe control actions can occur [2]. For each hazardous control action identified in the first step (i.e. those listed in **table 2**, **3**, **4**, **5**, **6** and **7**) one needs to *examine each part of their control loop to determine if that part could cause or contribute to the hazardous control action*. STPA provides guidance through general *casual factors* defined for the various parts of a generic control loop (see **Figure 5**).

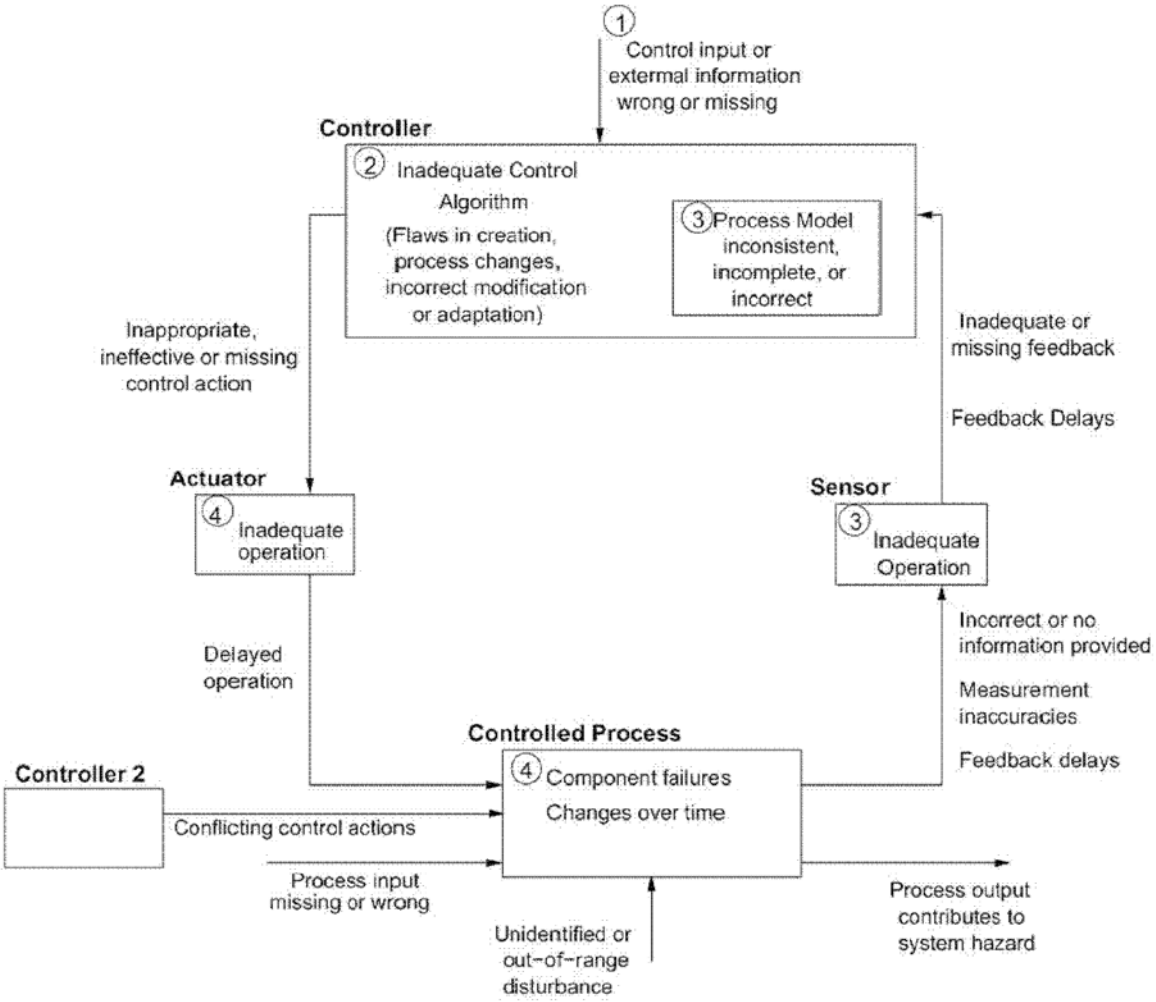

FIGURE 5 THE CAUSAL FACTORS TO BE CONSIDERED IN STEP 2 (SEE [2])

In subchapter 2.2 I defined "safe" tool integration software as that which does not by itself contribute to the accumulation of defects that can violate safety constraints in the end product. Translating this into a control loop applicable to our process models (i.e. **Figure 2**, **3** and **4**) puts tool integration mechanisms (or operators[11]) in the place of the controller and the end product in the place of the controlled process. The connection between these across a development process is the development artifacts (as actuators) and verification and validation activities (as sensors). The general STPA casual factors can thus be translated into specific ones for our context (see **Figure 6**).

[11] One may encounter any combination of assignment of tool integration mechanisms and operators to the roles of control input and controller (a tool integration mechanism may control an operator, etc.)

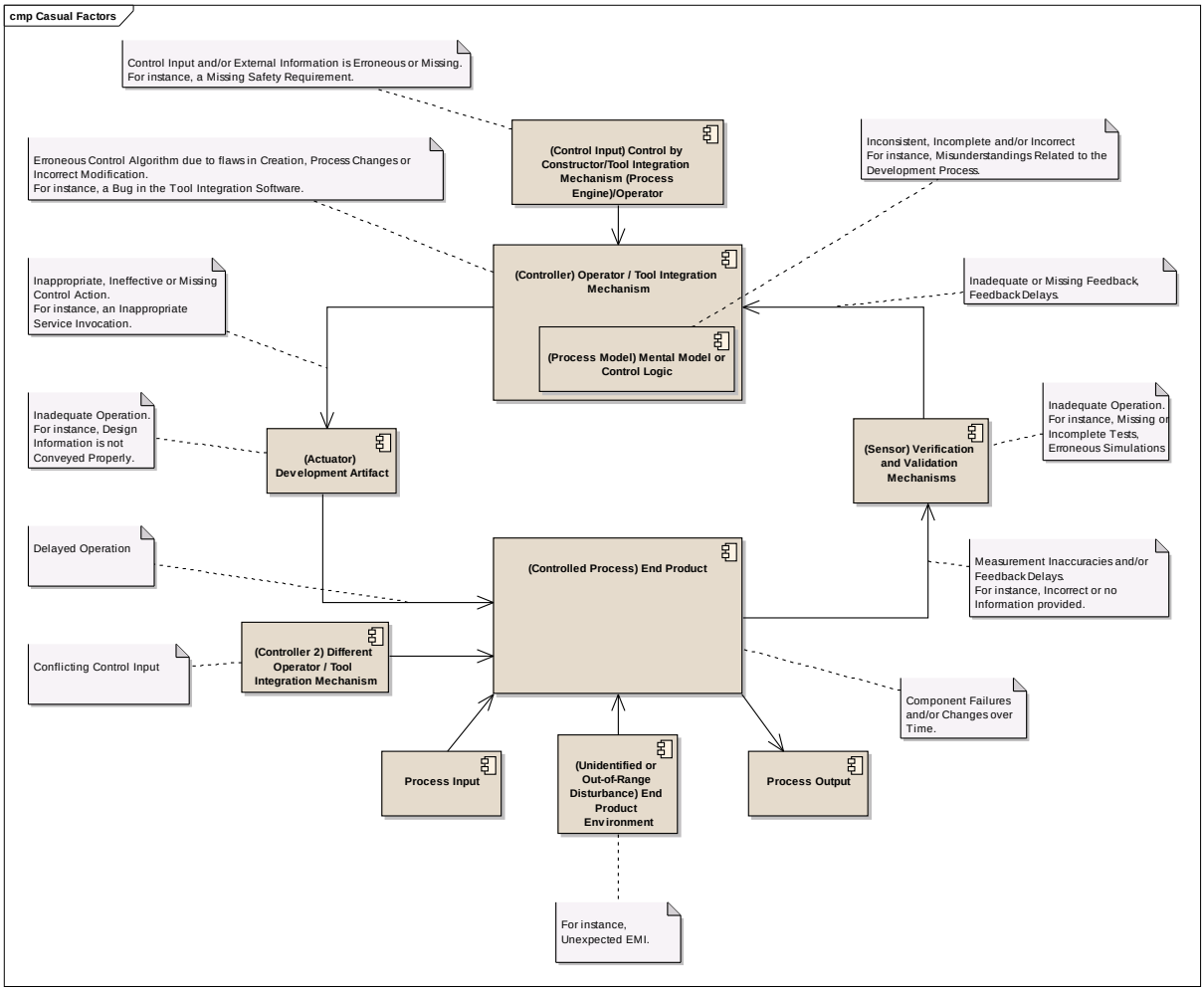

FIGURE 6 CASUAL FACTORS IN THE CONTEXT OF TOOL INTEGRATION

Based on the casual factors specific to the context of tool integration the causes for the risks listed in **Table 2** trough **Table 7** can be identified. I have listed them in **Table 8**, **Table 9** and **Table 10** (defining the causes of the risks in the original toolchain, the differences due to the changes in the SITC and differences due to the changes in the TITC, respectively).

I don´t consider the last part of the second step of a STPA [2], i.e. the way controls can degrade over time. This is a limitation of this work at this time and depending on the feedback on this technical report further work may proceed in that direction

Casual Risks	Causes
(1) Customer requirements are not correctly reproduced.	Customer does not understand his own requirements and provides erroneous information (misunderstands the end product environment, etc.). Product Manager does not understand Customers requirements (lack of training, not present in earlier projects, tight time plan, etc.). Product Manager enters erroneous information in the PRS (large amount of information in unstructured text, etc.). Product Manager enters erroneously structured information in the PRS (wrong format, wrong table style, etc.). PRS is corrupted during storage. Customer does not understand end product environment completely (unknown EMI, etc.). Management Team fails to understand the PRS completely (large amount of information in unstructured text, etc.) and accepts an erroneous PRS. Information from the Verification and Validation activities are not entered into the PRS and out-dated/erroneous information is used in new projects. Verification and Validation activities take too long time, information is not entered in time into the PRS and out-dated/erroneous information is used in new projects.
(2) Customer requirements are not completely captured.	Customer does not understand his own requirements and does not provide the whole picture (deems information irrelevant, etc.). Product Manager does not understand Customers requirements (fuzzy requirements, tight time plan, etc.). Product Manager enters incomplete information in the PRS (unclear where information should be entered, etc.). PRS is corrupted during storage. Customer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). Management Team fails to understand the PRS completely (fuzzy requirements, etc.) and accepts an incomplete PRS. Information from the Verification and Validation activities are not entered into the PRS and out-dated/erroneous information is used in new projects. Verification and Validation activities take too long time, information is not entered in time into the PRS and out-dated/erroneous information is used in new projects.
(3) Customer requirements are not correctly reproduced.	See (1) (with review by Engineers)
(4) Customer requirements are not completely captured.	See (2) (with review by Engineers)
(5) PRS and/or TS not correctly reproduced.	System Architect does not understand PRS and/or TS (lack of training, tight time plan, etc.). System Architect enters erroneous information in the SDS (large amount of information in unstructured text, etc.). Systems Architect enters erroneously structured information in the PRS (wrong format, wrong table style, etc.). SDS is corrupted during storage. System Architect does not understand end product environment completely (unknown EMI, etc.). No feedback on SDS from later development phases.
(6) PRS and/or TS not completely captured.	Unclear requirements on exactly what should be in the SDS and information is not entered (deemed irrelevant, etc.). System Architect does not understand PRS and/or TS (fuzzy requirements, tight time plan, etc.). System Architect enters incomplete information in the SDS (unclear where information should be entered, etc.). SDS is corrupted during storage. System Architect does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on SDS from later development phases.
(7) PRS and/or TS not correctly reproduced.	System Architect does not understand PRS and/or TS (lack of training, tight time plan, etc.). Systems Architect enters erroneously structured information in the MSSM (wrong format, etc.). MSSM is corrupted during storage. System Architect does not understand end product environment completely (unknown EMI, etc.). No feedback on MSSM from later development phases.
(8) PRS and/or TS not completely captured.	Unclear requirements on exactly what should be in the MSSM and information is not entered (deemed irrelevant, etc.). System Architect does not understand PRS and/or TS (fuzzy requirements, tight time plan, etc.). MSSM is corrupted during storage. System Architect does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on MSSM from later development phases.
(9) PRS and/or TS not correctly reproduced.	See (5)
(10) PRS and/or TS not completely captured.	See (6)
(11) PRS and/or TS not correctly reproduced.	See (5)
(12) PRS and/or TS not completely captured.	See (6)
(13) SDS and/or MSSM not correctly reproduced.	Matlab/Simulink Application Designer does not understand SDS and/or MSSM (lack of training, tight time plan, etc.). Matlab/Simulink Application Designer enters erroneous information in the M/SAD (large amount of information in unstructured text, etc.). Matlab/Simulink Application Designer enters erroneously structured information in the M/SAD (wrong format, wrong table style, etc.). M/SAD is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). No feedback on M/SAD from later development phases.
(14) SDS and/or MSSM model not completely captured.	Unclear requirements on exactly what should be in the M/SAD and information is not entered (deemed irrelevant, etc.). Matlab/Simulink Application Designer does not understand SDS and/or MSSM (fuzzy requirements, tight time plan, etc.). Matlab/Simulink Application Designer enters incomplete information in the SDS (unclear where information should be entered, etc.). M/SAD is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on M/SAD from later development phases.
(15) FRS and/or CHS not correctly reproduced.	FPGA Application Engineer does not understand FRS and/or CHS (lack of training, tight time plan, etc.). FPGA Application Engineer enters erroneous information in the FDS (large amount of information in unstructured text, etc.). FPGA Application Engineer enters erroneously structured information in the FDS (wrong format, wrong table style, etc.).

	FDS is corrupted during storage. FPGA Application Engineer does not understand end product environment completely (unknown EMI, etc.). No feedback on FDS from later development phases.
(16) FRS and/or CHS not completely captured.	Unclear requirements on exactly what should be in the FDS and information is not entered (deemed irrelevant, etc.). FPGA Application Engineer does not understand FRS and/or CHS (fuzzy requirements, tight time plan, etc.). FPGA Application Engineer enters incomplete information in the FDS (unclear where information should be entered, etc.). FDS is corrupted during storage. FPGA Application Engineer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on FDS from later development phases.
(17) FDS not correctly reproduced.	Matlab/Simulink Application Designer does not understand FDS (lack of training, tight time plan, etc.). Matlab/Simulink Application Designer enters erroneous information in the FIOL (large amount of information in unstructured text, etc.). Matlab/Simulink Application Designer enters erroneously structured information in the FIOL (wrong format, wrong table style, etc.). FIOL is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). FPGA Application Engineer fails to understand the FIOL completely (large amount of information in unstructured text, etc.) and accepts an erroneous FIOL. No feedback on FIOL from later development phases.
(18) FDS not completely captured.	Unclear requirements on exactly what should be in the FIOL and information is not entered (deemed irrelevant, etc.). Matlab/Simulink Application Designer does not understand FDS (fuzzy requirements, tight time plan, etc.). Matlab/Simulink Application Designer enters incomplete information in the FDS (unclear where information should be entered, etc.). FIOL is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). FPGA Application Engineer fails to understand the FIOL completely (fuzzy requirements, etc.) and accepts an incomplete FIOL. No feedback on FIOL from later development phases.
(19) SDS, FRS and/or MTM not correctly reproduced.	CIT Application Engineer does not understand SDS, FRS and/or MTM (lack of training, tight time plan, etc.). CIT Application Engineer enters erroneously structured information in the CA (wrong format, etc.). CA is corrupted during storage. CIT Application Engineer does not understand end product environment completely (unknown EMI, etc.). No feedback on CA from later development phases.
(20) SDS, FRS and/or MTM not completely captured.	Unclear requirements on exactly what should be in the CA and information is not entered (deemed irrelevant, etc.). CIT Application Engineer does not understand SDS, FRS and/or MTM (fuzzy requirements, tight time plan, etc.). CA is corrupted during storage. CIT Application Engineer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on CA from later development phases.
(21) FIOL, FDS and/or M/SAD not correctly reproduced.	Matlab/Simulink Application Designer does not understand FIOL, FDS and/or M/SAD (lack of training, tight time plan, etc.). Matlab/Simulink Application Designer enters erroneously structured information in the MTM (wrong format, etc.). MTM is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). No feedback on MTM from later development phases.
(22) FIOL, FDS and/or M/SAD not completely captured.	Unclear requirements on exactly what should be in the MTM and information is not entered (deemed irrelevant, etc.). Matlab/Simulink Application Designer does not understand FIOL, FDS and/or M/SAD (fuzzy requirements, tight time plan, etc.). MTM is corrupted during storage. Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). No feedback on MTM from later development phases.
(23) FDS not correctly reproduced.	FPGA Application Engineer does not understand FDS (lack of training, tight time plan, etc.). FPGA Application Engineer enters erroneously structured information in the FPGA Content (wrong format, etc.). FPGA Content corrupted during storage. FPGA Application Engineer does not understand end product environment completely (unknown EMI, etc.). V&V through simulation is erroneous (suboptimal optimization, etc.) and an erroneous FPGA Content is accepted.
(24) FDS not completely captured.	Unclear requirements on exactly what should be in the FPGA Content and information is not entered (deemed irrelevant, etc.). FPGA Application Engineer does not understand FDS (fuzzy requirements, tight time plan, etc.). FPGA Content is corrupted during storage. FPGA Application Engineer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). V&V through simulation is erroneous (suboptimal optimization, etc.) and an incomplete FPGA Content is accepted.
(25) MSSM is created incomplete.	System Architect not properly trained in Matlab. Matlab GUI does not allow for experimentation / give understandable feedback to a non-expert.
(26) MSSM is created erroneous.	System Architect not properly trained in Matlab. Matlab GUI does not allow for experimentation / give understandable feedback to a non-expert.
(27) CIT Application is created incomplete.	CIT Application Engineer not properly trained in Matlab. Matlab GUI does not allow for experimentation / give understandable feedback to a non-expert.
(28) CIT Application is created erroneous.	CIT Application Engineer not properly trained in Matlab. Matlab GUI does not allow for experimentation / give understandable feedback to a non-expert.
(29) Control Application is created erroneous (erroneously configured)	Matlab/Simulink Application Designer not properly trained in Windriver Workbench. Windriver Workbench GUI does not allow for experimentation / give understandable feedback to a non-expert.
(30) Control Application is created erroneous	Matlab/Simulink Application Designer not properly trained in GNU Compiler. GNU Compiler GUI does not allow for experimentation / give understandable feedback to a non-expert.

(erroneously configured)	
(31) Customer provides erroneous information and data state regress.	No or insufficient verification of requirements by customer prior to their release. Customer does not understand his own requirements and provides erroneous information (misunderstands the end product environment, etc.). Product Manager does not analyze properly or give proper feedback on requirements.
(32) Requirements from Old Projects contradict New Requirements, without this being detected.	No clear directive on which information should be extracted from Old Projects. Requirements handled in an informal way (large amount of information in unstructured text, etc.). Requirements are not updated throughout the project and obsolete information is kept active.
(33) Product Manager decides to continue with creating the PRS and TS before the Customer Requirements are fully understood	No clear directive on when it is acceptable to start creating the PRS and TS. Product Manager does not understand the implications of the Customer Requirements (lack of training, not present in earlier projects, tight time plan, etc.). Requirements handled in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that enough information has been obtained. Product Manager does not receive feedback on the state of the initial PRS and TS as submitted in Old Projects.
(34) Product Manager misinterprets Requirements from Old Projects and data state regress.	Product Manager does not understand the implications of the Old Customer Requirements (lack of training, not present in earlier projects, tight time plan, etc.). Requirements handled in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that enough information has been understood
(35) Data to the PRS is corrupted during manual transfer.	No clear directive that the PRS should be protected against data corruption. No clear directive on how the PRS should be protected against data corruption. Product Manager does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. Product Manager or Management Team is not informed about a new version of late, additional customer requirements.
(36) Data to the TS is corrupted during manual transfer.	No clear directive that the TS should be protected against data corruption. No clear directive on how the TS should be protected against data corruption. Product Manager does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. Product Manager or Engineers are not informed about a new version of late, additional customer requirements.
(37) The state of the PRS is not sufficient, but it is accepted.	No clear directive on when the PRS is acceptable. Management Team does not understand the implications of the PRS (lack of training, not present in earlier projects, tight time plan, etc.). PRS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the PRS is complete and correct. Management Team does not receive feedback on the state of the PRS as submitted in Old Projects.
(38) Management Team gives erroneous feedback and PRS state regress.	Management Team does not understand the implications of the PRS (lack of training, not present in earlier projects, tight time plan, etc.). PRS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding what the PRS states. Product Manager cannot comment on feedback from the Management Team, making this feedback always the final truth.
(39) The state of the TS is not sufficient, but it is accepted.	No clear directive on when the TS is acceptable. Engineers do not understand the implications of the TS (lack of training, not present in earlier projects, tight time plan, etc.). TS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the TS is complete and correct. Engineers do not receive feedback on the state of the TS as submitted in Old Projects.
(40) Engineers give erroneous feedback and TS state regress.	Engineers do not understand the implications of the TS (lack of training, not present in earlier projects, tight time plan, etc.). TS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding what the TS states. Product Manager cannot comment on feedback from the Engineers, making this feedback always the final truth.
(41) The PRS and the TS contradict each other.	No clear directive on how the PRS and TS should interrelate. Product Manager does not understand the implications of the interrelationships between the PRS and the TS (lack of training, tight time plan, etc.). 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. Different people review the different artifacts (both directly and through creating new documents based on the PRS and TS). No common feedback given to the Product Manager on both the PRS and the TS.
(42) Data to the SDS is corrupted during manual transfer.	No clear directive that the SDS should be protected against data corruption. No clear directive on how the SDS should be protected against data corruption. System Architect does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. System Architect is not informed about a new version of the PRS and/or TS.
(43) Data to the MSSM is corrupted during manual transfer.	No clear directive that the MSSM should be protected against data corruption. No clear directive on how the MSSM should be protected against data corruption. System Architect does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. System Architect is not informed about a new version of the PRS and/or TS.
(44) Data to the FRS is corrupted during manual transfer.	No clear directive that the FRS should be protected against data corruption. No clear directive on how the FRS should be protected against data corruption. System Architect does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. System Architect is not informed about a new version of the PRS and/or TS.
(45) Data to the CHS is corrupted during manual transfer.	No clear directive that the CHS should be protected against data corruption. No clear directive on how the CHS should be protected against data corruption. System Architect does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. System Architect is not informed about a new version of the PRS and/or TS.

(46) The state of the SDS is not sufficient, but it is used.	No clear directive on when the SDS is acceptable. System Architect and/or Matlab/Simulink Application Designer do not understand the implications of the SDS (lack of training, not present in earlier projects, tight time plan, etc.). SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct. The Matlab/Simulink Application Designer has no possibility to give feedback on the SDS to the System Architect. The System Architect does not receive feedback on the submission of an erroneous and/or incomplete SDS, when this shows up in V&V activities (testing, etc.).
(47) The state of the MSSM is not sufficient, but it is used.	No clear directive on when the MSSM is acceptable. System Architect and/or Matlab/Simulink Application Designer do not understand the implications of the MSSM (lack of training, not present in earlier projects, tight time plan, etc.). The Matlab/Simulink Application Designer has no possibility to give feedback on the MSSM to the System Architect. The System Architect does not receive feedback on the submission of an erroneous and/or incomplete MSSM, when this shows up in V&V activities (testing, etc.)
(48) The state of the FRS is not sufficient, but it is used.	No clear directive on when the FRS is acceptable. System Architect and/or FPGA Application Engineer do not understand the implications of the FRS (lack of training, not present in earlier projects, tight time plan, etc.). FRS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct. The FPGA Application Engineer has no possibility to give feedback on the FRS to the System Architect. The System Architect does not receive feedback on the submission of an erroneous and/or incomplete FRS, when this shows up in V&V activities (testing, etc.)
(49) The state of the CHS is not sufficient, but it is used.	No clear directive on when the CHS is acceptable. System Architect and/or FPGA Application Engineer do not understand the implications of the CHS (lack of training, not present in earlier projects, tight time plan, etc.). CHS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct. The FPGA Application Engineer has no possibility to give feedback on the CHS to the System Architect. The System Architect does not receive feedback on the submission of an erroneous and/or incomplete CHS, when this shows up in V&V activities (testing, etc.)
(50) The SDS and the MSSM contradict each other.	No clear directive on how the SDS and MSSM should interrelate. Matlab/Simulink Application Designer does not understand the implications of the interrelationships between the SDS and the MSSM (lack of training, tight time plan, etc.). SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the SDS. SDS and MSSM structured in different ways (text vs. formal structure, etc.), leading to misunderstandings regarding the coverage of the artifacts. No common feedback given to the System Architect on both the SDS and the MSSM.
(51) Data to the M/SAD is corrupted during manual transfer.	No clear directive that the M/SAD should be protected against data corruption. No clear directive on how the M/SAD should be protected against data corruption. Matlab/Simulink Application Designer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. Matlab/Simulink Application Designer is not informed about a new version of the SDS and/or MSSM.
(52) The FRS and the CHS contradict each other.	No clear directive on how the FRS and CHS should interrelate. FPGA Application Engineer does not understand the implications of the interrelationships between the FRS and the CHS (lack of training, tight time plan, etc.). FRS and CHS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the FRS and/or CHS. No common feedback given to the System Architect on both the FRS and the CHS.
(53) Data to the FDS is corrupted during manual transfer.	No clear directive that the FDS should be protected against data corruption. No clear directive on how the FDS should be protected against data corruption. FPGA Application Engineer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. FPGA Application Engineer is not informed about a new version of the FRS and/or CHS.
(54) The state of the M/SAD is not sufficient, but it is used.	No clear directive on when the M/SAD is acceptable. Matlab/Simulink Application Designer does not understand the implications of the M/SAD (lack of training, not present in earlier projects, tight time plan, etc.). No-one gives feedback on the M/SAD to the Matlab/Simulink Application Designer. The Matlab/Simulink Application Designer does not receive feedback on the submission of an erroneous and/or incomplete M/SAD, when this shows up in V&V activities (testing, etc.).
(55) The state of the FDS is not sufficient, but it is used.	No clear directive on when the FDS is acceptable. FPGA Application Engineer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). No-one gives feedback on the FDS to the FPGA Application Engineer. The FPGA Application Engineer does not receive feedback on the submission of an erroneous and/or incomplete FDS, when this shows up in V&V activities (testing, etc.).
(56) The state of the FDS is not sufficient, but it is used.	No clear directive on when the FDS is acceptable. FPGA Application Engineer and/or Matlab/Simulink Application Designer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). FDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FDS is complete and correct. The Matlab/Simulink Application Designer has no possibility to give feedback on the FDS to the FPGA Application Engineer. The FPGA Application Engineer does not receive feedback on the submission of an erroneous and/or incomplete FDS, when this shows up in V&V activities (testing, etc.).
(57) The state of the FDS is not sufficient, but it is used.	No clear directive on when the FDS is acceptable. FPGA Application Engineer and/or Matlab/Simulink Application Designer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). FDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FDS is complete and correct. The Matlab/Simulink Application Designer has no possibility to give feedback on the FDS to the FPGA Application Engineer. The FPGA Application Engineer does not receive feedback on the submission of an erroneous and/or incomplete FDS, when this shows up in V&V activities (testing, etc.).
(58) Data to the FIOL is corrupted during manual transfer.	No clear directive that the FIOL should be protected against data corruption. No clear directive on how the FIOL should be protected against data corruption. FPGA Application Engineer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. Matlab/Simulink Application Designer is not informed about a new version of the FDS.
(59) The state of the FIOL is not sufficient, but it is accepted.	No clear directive on when the FIOL is acceptable. FPGA Application Engineer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FIOL is complete and correct. FPGA Application Engineer does not receive feedback on the state of the FIOL as submitted in Old Projects.

	Matlab/Simulink Application Designer does not receive feedback on the submission of an erroneous and/or incomplete FIOL, when this shows up in V&V activities (testing, etc.)
(60) FPGA Application Engineer gives erroneous feedback and FIOL state regress.	FPGA Application Engineer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding what the FIOL states. Matlab/Simulink Application Designer cannot comment on feedback from the FPGA Application Engineer, making this feedback always the final truth.
(61) Data to the FC is corrupted during manual transfer.	No clear directive that the FC should be protected against data corruption. No clear directive on how the FC should be protected against data corruption. FPGA Application Engineer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. FPGA Application Engineer is not informed about a new version of the FDS.
(62) The state of the FIOL is not sufficient, but it is used.	No clear directive on when the FIOL is acceptable. Matlab/Simulink Application Designer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FIOL is complete and correct.
(63) The state of the FC is not sufficient, but it is accepted.	No clear directive on when the FC is acceptable. FPGA Application Engineer does not understand the implications of the FC (lack of training, not present in earlier projects, tight time plan, etc.). No-one gives feedback on the FC to the FPGA Application Engineer. The FPGA Application Engineer does not receive feedback on the submission of an erroneous and/or incomplete FC, when this shows up in V&V activities (testing, etc.).
(64) The FDS, the M/SAD and the FIOL contradict each other.	No clear directive on how the FDS, M/SAD and FIOL should interrelate. Matlab/Simulink Application Designer does not understand the implications of the interrelationships between the FDS, M/SAD and FIOL (lack of training, tight time plan, etc.). FDS, M/SAD and FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the different artifacts. Different people review the different artifacts (both directly and through creating new artifacts based on the FDS and M/SAD). No common feedback given to the FPGA Application Engineer and/or the Matlab/Simulink Application Designer on the FDS, M/SAD and FIOL.
(65) The FDS is interpreted differently in different process branches.	No clear directive on how ambiguities in the FDS should be handled. The Matlab/Simulink Application Designer and the FPGA Application Engineer have little insight in each other´s domains (lack of training, tight time plan, etc.). FDS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the contents of the FDS. Different people review the FDS at different times without communicating. The Matlab/Simulink Application Designer and/or the FPGA Application Engineer are not informed about a new version of the FDS. Different V&V activities lead to updates in one domain, but not in others, since the information is not entered into the FDS.
(66) Data to the MTM is corrupted during manual transfer.	No clear directive that the MTM should be protected against data corruption. No clear directive on how the MTM should be protected against data corruption. Matlab/Simulink Application Designer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. Matlab/Simulink Application Designer is not informed about a new version of the M/SAD, FDS and/or FIOL.
(67) The state of the MTM is not sufficient, but it is used.	No clear directive on when the MTM is acceptable. Matlab/Simulink Application Designer does not understand the implications of the MTM (lack of training, not present in earlier projects, tight time plan, etc.).
(68) The state of the SDS is not sufficient, but it is used.	No clear directive on when the SDS is acceptable. CIT Application Engineer does not understand the implications of the SDS (lack of training, not present in earlier projects, tight time plan, etc.). SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct.
(69) The state of the FRS is not sufficient, but it is used.	No clear directive on when the FRS is acceptable. CIT Application Engineer does not understand the implications of the FRS (lack of training, not present in earlier projects, tight time plan, etc.). FRS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FRS is complete and correct.
(70) The state of the MTM is not sufficient, but it is used.	No clear directive on when the MTM is acceptable. CIT Application Engineer does not understand the implications of the MTM (lack of training, not present in earlier projects, tight time plan, etc.).
(71) The SDS, the FRS and the MTM contradict each other.	No clear directive on how the SDS, FRS and MTM should interrelate. CIT Application Engineer does not understand the implications of the interrelationships between the SDS, FRS and MTM (lack of training, tight time plan, etc.). SDS and FRS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the different artifacts. Different people review the different artifacts (both directly and through creating new artifacts based on the SDS, FRS and MTM). No common feedback given to the FPGA Application Engineer and/or the Matlab/Simulink Application Designer on the SDS, FRS and/or MTM.
(72) The SDS is interpreted differently in different process branches.	No clear directive on how ambiguities in the SDS should be handled. The Matlab/Simulink Application Designer and the CIT Application Engineer have little insight in each other´s domains (lack of training, tight time plan, etc.). SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the contents of the SDS. Different people review the SDS at different times without communicating. The Matlab/Simulink Application Designer and/or the CIT Application Engineer are not informed about a new version of the SDS. Different V&V activities lead to updates in one domain, but not in others, since the information is not entered into the SDS.
(73) The FRS is interpreted differently in different process branches.	No clear directive on how ambiguities in the FRS should be handled. The Matlab/Simulink Application Designer, the FPGA Application Engineer and the CIT Application Engineer have little insight in each other´s domains (lack of training, tight time plan, etc.). FRS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the contents of the FRS. Different people review the FRS at different times without communicating. The Matlab/Simulink Application Designer, the FPGA Application Engineer and/or the CIT Application Engineer are not informed about a new version of the FRS. Different V&V activities lead to updates in one domain, but not in others, since the information is not entered into the FRS.
(74) Data to the CA is corrupted during manual transfer.	No clear directive that the CA should be protected against data corruption. No clear directive on how the CA should be protected against data corruption. CIT Application Engineer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). Data Integrity is not ensured by Platform. CIT Application Engineer is not informed about a new version of the SDS, FDS and/or FIOL.

(75) To create the SDS takes too long time, due to manual transformation of data. An incomplete version of the SDS is therefore used.	No clear directive on when the SDS is acceptable. System Architect does not understand the implications of the SDS (lack of training, not present in earlier projects, tight time plan, etc.). SDS is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the SDS prematurely due to project timing constraints. PRS and/or TS are not structured efficiently and/or appropriately. Late changes in the PRS and/or TS require extensive rework of the SDS. Late feedback on the SDS from the Matlab/Simulink Application Designer and/or CIT Application Engineer requires extensive rework of the SDS.
(76) To create the MSSM takes too long time, due to manual transformation of data. An incomplete version of the MSSM is therefore used.	No clear directive on when the MSSM is acceptable. System Architect does not understand the implications of the MSSM (lack of training, not present in earlier projects, tight time plan, etc.). MSSM is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the MSSM prematurely due to project timing constraints. PRS and/or TS are not structured efficiently and/or appropriately. Late changes in the PRS and/or TS require extensive rework of the MSSM. Late feedback on the MSSM from the Matlab/Simulink Application Designer and/or CIT Application Engineer requires extensive rework of the MSSM.
(77) To create the FRS takes too long time, due to manual transformation of data. An incomplete version of the FRS is therefore used.	No clear directive on when the FRS is acceptable. System Architect does not understand the implications of the FRS (lack of training, not present in earlier projects, tight time plan, etc.). FRS is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the FRS prematurely due to project timing constraints. PRS and/or TS are not structured efficiently and/or appropriately. Late changes in the PRS and/or TS require extensive rework of the FRS. Late feedback on the FRS from the FPGA Application Engineer and/or CIT Application Engineer requires extensive rework of the FRS.
(78) To create the CHS takes too long time, due to manual transformation of data. An incomplete version of the CHS is therefore used.	No clear directive on when the CHS is acceptable. System Architect does not understand the implications of the CHS (lack of training, not present in earlier projects, tight time plan, etc.). CHS is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the CHS prematurely due to project timing constraints. PRS and/or TS are not structured efficiently and/or appropriately. Late changes in the PRS and/or TS require extensive rework of the CHS. Late feedback on the CHS from the FPGA Application Engineer requires extensive rework of the CHS.
(79) To create the M/SAD takes too long time, due to manual transformation of data. An incomplete version of the M/SAD is therefore used.	No clear directive on when the M/SAD is acceptable. Matlab/Simulink Application Designer does not understand the implications of the M/SAD (lack of training, not present in earlier projects, tight time plan, etc.). M/SAD is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the M/SAD prematurely due to project timing constraints. SDS and/or MSSM are not structured efficiently and/or appropriately. Late changes in the SDS and/or MSSM require extensive rework of the M/SAD. No feedback on the M/SAD leads to problems not being found until very late in the project, requiring extensive rework of the M/SAD.
(80) To create the FDS takes too long time, due to manual transformation of data. An incomplete version of the FDS is therefore used.	No clear directive on when the FDS is acceptable. FPGA Application Engineer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). FDS is not structured efficiently and/or appropriately. Management Team enforces the acceptance of the FDS prematurely due to project timing constraints. FRS and/or CHS are not structured efficiently and/or appropriately. Late changes in the FRS and/or CHS require extensive rework of the FDS. Late feedback on the FDS from the Matlab/Simulink Application Designer requires extensive rework of the FDS.
(81) To create the FIOL takes too long time, due to manual transformation of data. An incomplete version of the FIOL is therefore used.	No clear directive on when the FIOL is acceptable. Matlab/Simulink Application Designer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). FIOL is not structured efficiently and/or appropriately. FPGA Application Engineer enforces the acceptance of the FIOL prematurely due to project timing constraints. FDS is not structured efficiently and/or appropriately. Late changes in the FDS require extensive rework of the FIOL. Late feedback on the FIOL from the FPGA Application Engineer requires extensive rework of the FIOL.
(82) To create the FC takes too long time, due to manual transformation of data. An incomplete version of the FC is therefore used.	No clear directive on when the FC is acceptable. FPGA Application Engineer does not understand the implications of the FC (lack of training, not present in earlier projects, tight time plan, etc.). Management Team enforces the acceptance of the FC prematurely due to project timing constraints. FDS is not structured efficiently and/or appropriately. Late changes in the FDS require extensive rework of the FC.
(83) Simulation ended too soon, due to manual decision. A suboptimal/erroneous version of the FC is therefore used.	No clear directive on when the FC is acceptable. FPGA Application Engineer does not understand the implications of the FC (lack of training, not present in earlier projects, tight time plan, etc.). FPGA Application Engineer does not understand the implications of the Simulation Results (lack of training, not present in earlier projects, tight time plan, etc.). Management Team enforces the acceptance of the FC prematurely due to project timing constraints. FPGA Application Engineer does not receive feedback on the submission of an erroneous and/or incomplete FC, when this shows up in V&V activities (testing, etc.)
(84) To create the MTM takes too long time, due to manual transformation of data. An incomplete version of the MTM is therefore used.	No clear directive on when the MTM is acceptable. Matlab/Simulink Application Designer does not understand the implications of the MTM (lack of training, not present in earlier projects, tight time plan, etc.). Management Team enforces the acceptance of the MTM prematurely due to project timing constraints. M/SAD, FDS and/or FIOL are not structured efficiently and/or appropriately. Late changes in the M/SAD, FDS and/or FIOL require extensive rework of the MTM.
(85) To create the CA takes too long time, due	No clear directive on when the CA is acceptable.

to manual transformation of data. An incomplete version of the CA is therefore used.	CIT Application Engineer does not understand the implications of the CA (lack of training, not present in earlier projects, tight time plan, etc.). Management Team enforces the acceptance of the CA prematurely due to project timing constraints. SDS and/or FRS are not structured efficiently and/or appropriately. Late changes in the SDS, FRS and/or MTM require extensive rework of the CA.

TABLE 8 CAUSES OF THE RISKS IN THE ORIGINAL TOOLCHAIN

Casual Risks	Causes
(13) SDS ~~and/or MSSM~~ not correctly reproduced.	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[12]: • Matlab/Simulink Application Designer does not understand the MSSM (lack of training, tight time plan, etc.). The following risks are mitigated[13] by introducing automated transformations from the MSSM to the M/SAD: • Matlab/Simulink Application Designer enters erroneous information in the M/SAD (large amount of information in unstructured text, etc.). • Matlab/Simulink Application Designer enters erroneously structured information in the M/SAD (wrong format, wrong table style, etc.). The following risk is mitigated by data integration creating the possibility[14] to ensure Data Integrity throughout the development process: • M/SAD is corrupted during storage. The following risks are mitigated by data integration creating the possibility[15] for traceability to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). • No feedback on M/SAD from later development phases. Based on this I can conclude that parts of this particular casual risk is no longer probable in our context.
~~(15) FRS and/or CHS not correctly reproduced.~~	See (13) above[16].
(21) FIOL~~, FDS and/or M/SAD~~ not correctly reproduced.	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[17]: • Matlab/Simulink Application Designer does not understand FDS and/or M/SAD (lack of training, tight time plan, etc.). The following risk is mitigated by introducing automated transformations from the FDS and M/SAD to the MTM: • Matlab/Simulink Application Designer enters erroneously structured information in the MTM (wrong format, etc.). The following risk is mitigated by data integration creating the possibility to ensure Data Integrity throughout the development process: • MTM is corrupted during storage. The following risks are mitigated by data integration creating the possibility for traceability to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). • No feedback on MTM from later development phases. Based on this I can conclude that parts of this particular casual risk is no longer probable in our context.
(22) FIOL~~, FDS and/or M/SAD~~ not completely captured.	The following risks are mitigated by introducing a formal way of representing the FDS and the M/SAD: • Unclear requirements on exactly what should be in the MTM (in regard to the FDS and M/SAD) and information is not entered (deemed irrelevant, etc.). • Matlab/Simulink Application Designer does not understand FDS and/or M/SAD (fuzzy requirements, tight time plan, etc.). The following risk is mitigated by data integration creating the possibility to ensure Data Integrity throughout the development process: • MTM is corrupted during storage. The following risks are mitigated by data integration creating the possibility for traceability to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). • No feedback on MTM from later development phases. Based on this I can conclude that parts of this particular casual risk is no longer probable in our context.
~~(23) FDS not correctly reproduced.~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[18]: • FPGA Application Engineer does not understand FDS (lack of training, tight time plan, etc.). The following risk is mitigated by introducing automated transformations from the FDS to the FC: • FPGA Application Engineer enters erroneously structured information in the FPGA Content (wrong format, etc.). The following risk is mitigated by data integration creating the possibility to ensure Data Integrity throughout the development process: • FPGA Content corrupted during storage. The following risks are mitigated by data integration creating the possibility for traceability to give feedback on consistency and completeness: • FPGA Application Engineer does not understand end product environment completely (unknown EMI, etc.). • V&V through simulation is erroneous (suboptimal optimization, etc.) and an erroneous FPGA Content is accepted.[19]

[12] It is transformed into the similar risks associated with (87), since there can be mistakes in how the data transformations are constructed due to insufficient knowledge on part of the Tool Chain Developers.
[13] If the transformations are correct, something which need to be sufficiently assured by software qualification efforts.
[14] The possibility still needs to be realized; see subchapter 9.2 and 9.3 for a discussion regarding problems surrounding this.
[15] The possibility still needs to be realized; see subchapter 9.2 and 9.3 for a discussion regarding problems surrounding this.
[16] The similar risks that are added are those associated with (89).
[17] The similar risks that are added are those associated with (92).
[18] The similar risks that are added are those associated with (89).

	Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(24) FDS not completely captured.~~	The following risks are mitigated by introducing a formal way of representing the FDS: • Unclear requirements on exactly what should be in the FPGA Content and information is not entered (deemed irrelevant, etc.). • FPGA Application Engineer does not understand FDS (fuzzy requirements, tight time plan, etc.). The following risk is mitigated by data integration creating the possibility to ensure Data Integrity throughout the development process: • FPGA Content is corrupted during storage. The following risks are mitigated by data integration creating the possibility for traceability to give feedback on consistency and completeness: • FPGA Application Engineer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). • V&V through simulation is erroneous (suboptimal optimization, etc.) and an incomplete FPGA Content is accepted. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(25) MSSM is created incomplete. (26) MSSM is created erroneous.~~ (27) CIT Application is created incomplete. (28) CIT Application is created erroneous.	MSSN now created using a generic modeling tool. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(61) Data to the FC is corrupted during manual transfer.~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[20]: • No clear directive that the FC should be protected against data corruption. • FPGA Application Engineer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). The following risk is mitigated by data integration creating the possibility to ensure Data Integrity throughout the development process: • No clear directive on how the FC should be protected against data corruption. • Data Integrity is not ensured by Platform. The following risk is mitigated by data integration creating the possibility[21] to build process integration (notifications and/or process control) throughout the development process: • FPGA Application Engineer is not informed about a new version of the FDS. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(82) To create the FC takes too long time, due to manual transformation of data. An incomplete version of the FC is therefore used.~~	This casual risk is not mitigated by data integration *per se*, but rather transformed into a different form[22].
(86) Required transformations to M/SAD not constructed prior to launch of tool chain.	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in the M/SAD structure require more transformations. No feedback on Tool Chain to System Engineer from Matlab/Simulink Application Designer.
(87) The M/SAD is created erroneous	The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the DIS implementation (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Matlab/Simulink Application Designer will actually use the DIS. No verification of DIS.
(88) Required transformations to FDS not constructed prior to launch of tool chain.	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in the FDS structure require more transformations. No feedback on Tool Chain to System Engineer from FPGA Application Engineer.
(89) The FDS is created erroneous	The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the DIS implementation (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the FPGA Application Engineer will actually use the DIS. No verification of DIS.
(90) Required transformations to MTM not constructed prior to launch of tool chain.	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in the MTM structure require more transformations. No feedback on Tool Chain to System Engineer from Matlab/Simulink Application Designer.
(91) Required transformations to Control	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers.

[19] This V&V activity would act to verify the system and therefore require traces to start at an equivalent level of abstraction (i.e. requirements).
[20] The similar risks that are added are those associated with (95).
[21] The possibility still needs to be realized; see subchapter 9.2 and 9.3 for a discussion regarding this.
[22] The casual risk that replaces this casual risk is (104).

Application not constructed prior to launch of tool chain.	Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in the Control Application structure require more transformations. No feedback on Tool Chain to System Engineer from Matlab/Simulink Application Designer
(92) The MTM is created erroneous	The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the DIS implementation (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Matlab/Simulink Application Designer will actually use the DIS. No verification of DIS.
(93) The Control Application is created erroneous	The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the DIS implementation (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Matlab/Simulink Application Designer will actually use the DIS. No verification of DIS.
(94) Required transformations to FC not constructed prior to launch of tool chain.	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in the FC structure require more transformations. No feedback on Tool Chain to System Engineer from FPGA Application Engineer.
(95) The FC is created erroneous	The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the DIS implementation (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the FPGA Application Engineer will actually use the DIS. No verification of DIS.
(96) M/SAD is created incomplete.	Matlab Application Designer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(97) M/SAD is created erroneous.	Matlab Application Designer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(98) CIT Application is created incomplete.	CIT Application Engineer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(99) CIT Application is created erroneous.	CIT Application Engineer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(100) FDS is created incomplete.	FPGA Application Engineer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(101) FDS is created erroneous.	FPGA Application Engineer not properly trained in IRQA. IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert.
(102) To create the M/SAD takes too long time, due to the data integration transformations. An incomplete version of the M/SAD is therefore used.	No clear directive on when the M/SAD is acceptable. Matlab/Simulink Application Designer does not understand the implications of the M/SAD (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of M/SAD. Feedback on M/SAD completion takes too long time.
(103) To create the FDS takes too long time, due to the data integration transformations. An incomplete version of the FDS is therefore used.	No clear directive on when the FDS is acceptable. FPGA Application Engineer does not understand the implications of the FDS (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of FDS. Feedback on FDS completion takes too long time.
(104) To create the FC takes too long time, due to the data integration transformations. An incomplete version of the FC is therefore used.	No clear directive on when the FC is acceptable. FPGA Application Engineer does not understand the implications of the FC (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of FC. Feedback on FC completion takes too long time.
(105) To create the MTM takes too long time, due to the data integration transformations. An incomplete version of the MTM is therefore used.	No clear directive on when the MTM is acceptable. Matlab/Simulink Application Designer does not understand the implications of the MTM (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of MTM. Feedback on MTM completion takes too long time.

TABLE 9 DIFFERENCES IN CAUSES OF RISKS DUE TO CHANGES IN THE SECOND ITERATION OF THE TOOL CHAIN

Casual Risks	Causes
~~(13) SDS not correctly reproduced.~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[23]: • Matlab/Simulink Application Designer does not understand the SDS (lack of training, tight time plan, etc.). The following risks are mitigated by introducing automated transformations from the SDS to the M/SAD: • Matlab/Simulink Application Designer enters erroneous information in the M/SAD (large amount of information in unstructured text, etc.). • Matlab/Simulink Application Designer enters erroneously structured information in the M/SAD (wrong format, wrong table style, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • M/SAD is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). • No feedback on M/SAD from later development phases. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(17) FDS not correctly reproduced.~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form: • Matlab/Simulink Application Designer does not understand FDS (lack of training, tight time plan, etc.). The following risks are mitigated by introducing automated transformations from the FDS to the FIOL: • Matlab/Simulink Application Designer enters erroneous information in the FIOL (large amount of information in unstructured text, etc.). • Matlab/Simulink Application Designer enters erroneously structured information in the FIOL (wrong format, wrong table style, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • FIOL is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). • No feedback on FIOL from later development phases. The following risks are mitigated by introducing a formal way of representing the FIOL: • FPGA Application Engineer fails to understand the FIOL completely (large amount of information in unstructured text, etc.) and accepts an erroneous FIOL. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(18) FDS not completely captured.~~	The following risks are mitigated by introducing a formal way of representing the FDS and the M/SAD: • Unclear requirements on exactly what should be in the FIOL and information is not entered (deemed irrelevant, etc.). • Matlab/Simulink Application Designer does not understand FDS (fuzzy requirements, tight time plan, etc.). • Matlab/Simulink Application Designer enters incomplete information in the FDS (unclear where information should be entered, etc.). • FPGA Application Engineer fails to understand the FIOL completely (fuzzy requirements, etc.) and accepts an incomplete FIOL. The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • FIOL is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). • No feedback on FIOL from later development phases. Based on this I can conclude that this particular casual risk is no longer probable in our context.
(19) SDS, FRS ~~and/or MTM~~ not correctly reproduced.	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form: • CIT Application Engineer does not understand the MTM (lack of training, tight time plan, etc.). The following risks are mitigated by introducing automated transformations from the MTM to the CA: • CIT Application Engineer enters erroneously structured information in the CA (wrong format, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • CA is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • CIT Application Engineer does not understand end product environment completely (unknown EMI, etc.). • No feedback on CA from later development phases. Based on this I can conclude that parts of this particular casual risk is no longer probable in our context.

[23] It is transformed into the similar risks associated with (109).

(20) SDS, FRS ~~and/or MTM~~ not completely captured.	The following risks are mitigated by introducing a formal way of representing the MTM: • Unclear requirements on exactly what should be in the CA and information is not entered (deemed irrelevant, etc.). • CIT Application Engineer does not understand the MTM (fuzzy requirements, tight time plan, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • CA is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • CIT Application Engineer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). • No feedback on CA from later development phases. Based on this I can conclude that parts of this particular casual risk is no longer probable in our context.
~~(21) FIOL not correctly reproduced.~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form: • Matlab/Simulink Application Designer does not understand the FIOL (lack of training, tight time plan, etc.). The following risks are mitigated by introducing automated transformations from the FIOL to the MTM: • Matlab/Simulink Application Designer enters erroneously structured information in the MTM (wrong format, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • MTM is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (unknown EMI, etc.). • No feedback on MTM from later development phases. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(22) FIOL not completely captured.~~	The following risks are mitigated by introducing a formal way of representing the FIOL: • Unclear requirements on exactly what should be in the MTM and information is not entered (deemed irrelevant, etc.). • Matlab/Simulink Application Designer does not understand the FIOL (fuzzy requirements, tight time plan, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • MTM is corrupted during storage. The following risks are mitigated by data integration creating traceability support to give feedback on consistency and completeness: • Matlab/Simulink Application Designer does not understand end product environment completely (not apparent that a requirement means that a specific environment is present, etc.). • No feedback on MTM from later development phases. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(27) CIT Application is created incomplete.~~	The following risks are mitigated by introducing automation (no extended interaction with the Matlab GUI by Non-Expert): • CIT Application Engineer not properly trained in Matlab. • Matlab GUI does not allow for experimentation / give understandable feedback to a non-expert. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(28) CIT Application is created erroneous.~~	See (27) above.
~~(29) Control Application is created erroneous (erroneously configured)~~	See (27) above.
~~(30) Control Application is created erroneous (erroneously configured)~~	See (27) above.
~~(46) The state of the SDS is not sufficient, but it is used~~	The following risks are mitigated by data integration creating Data Mining support to analyze the current state of the project and its artifacts: • No clear directive on when the SDS is acceptable. The following risks are mitigated by introducing a formal way of representing the SDS: • System Architect and/or Matlab/Simulink Application Designer do not understand the implications of the SDS (lack of training, not present in earlier projects, tight time plan, etc.). • SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct. The following risks are mitigated by process integration notification support: • The Matlab/Simulink Application Designer has no possibility to give feedback on the SDS to the System Architect. • The System Architect does not receive feedback on the submission of an erroneous and/or incomplete SDS, when this shows up in V&V activities (testing, etc.). Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(47) The state of the MSSM is not sufficient, but it is used~~	See (46) above.
~~(48) The state of the FRS is not sufficient, but it~~	See (46) above.

~~is used~~	
~~(49) The state of the CHS is not sufficient, but it is used~~	See (46) above.
~~(50) The SDS and the MSSM contradict each other~~	See (46) above.
~~(52) The FRS and the CHS contradict each other~~	The following risks are mitigated by introducing a formal way of representing the FRS and CHS: • No clear directive on how the FRS and CHS should interrelate. • FPGA Application Engineer does not understand the implications of the interrelationships between the FRS and the CHS (lack of training, tight time plan, etc.). • FRS and CHS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the FRS and/or CHS. The following risks are mitigated by process integration notification support: • No common feedback given to the System Architect on both the FRS and the CHS. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(54) The state of the M/SAD is not sufficient, but it is used~~	See (46) above.
~~(55) The state of the FDS is not sufficient, but it is used~~	See (46) above.
~~(56) The state of the FDS is not sufficient, but it is used~~	See (46) above.
~~(57) The state of the FDS is not sufficient, but it is used~~	See (46) above.
~~(58) Data to the FIOL is corrupted during manual transfer~~	The following risk is not mitigated by data integration *per se*, but rather transformed into a different form[24]: • No clear directive that the FIOL should be protected against data corruption. • Matlab/Simulink Application Designer does not understand how to ensure data integrity (lack of training, tight time plan, etc.). The following risk is mitigated by data integration ensuring Data Integrity throughout the development process: • No clear directive on how the FIOL should be protected against data corruption. • Data Integrity is not ensured by Platform. The following risk is mitigated by process integration notification support: • Matlab/Simulink Application Designer is not informed about a new version of the FIOL. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(59) The state of the FIOL is not sufficient, but it is accepted~~	The following risks are mitigated by data integration creating Data Mining support to analyze the current state of the project and its artifacts: • No clear directive on when the FIOL is acceptable. The following risks are mitigated by introducing a formal way of representing the FIOL: • FPGA Application Engineer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). • FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the FIOL is complete and correct. The following risks are mitigated by process integration notification support: • FPGA Application Engineer does not receive feedback on the state of the FIOL as submitted in Old Projects. • Matlab/Simulink Application Designer does not receive feedback on the submission of an erroneous and/or incomplete FIOL, when this shows up in V&V activities (testing, etc.) Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(62) The state of the FIOL is not sufficient, but it is used~~	See (59) above.
~~(63) The state of the FC is not sufficient, but it is accepted~~	See (59) above.
~~(64) The FDS, the M/SAD and the FIOL contradict each other~~	The following risks are mitigated by introducing a formal way of representing the FDS, M/SAD and FIOL: • No clear directive on how the FDS, M/SAD and FIOL should interrelate. • Matlab/Simulink Application Designer does not understand the implications of the interrelationships between the FDS, M/SAD and FIOL (lack of training, tight time plan, etc.). • FDS, M/SAD and FIOL structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the coverage of the different artifacts. The following risks are mitigated by process integration notification support: • Different people review the different artifacts (both directly and through creating new artifacts based on the FDS and M/SAD). • No common feedback given to the FPGA Application Engineer and/or the Matlab/Simulink Application Designer on the FDS, M/SAD and FIOL.

[24] It is transformed into the similar risks associated with (113).

	Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(65) The FDS is interpreted differently in different process branches~~	The following risks are mitigated by introducing a formal way of representing the FDS: • No clear directive on how ambiguities in the FDS should be handled. • FDS structured in an informal way (large amount of information in unstructured text, etc.), leading to misunderstandings regarding the contents of the FDS. The following risks are mitigated by process integration notification support: • The Matlab/Simulink Application Designer and the FPGA Application Engineer have little insight in each other´s domains (lack of training, tight time plan, etc.). • Different people review the FDS at different times without communicating. • The Matlab/Simulink Application Designer and/or the FPGA Application Engineer are not informed about a new version of the FDS. • Different V&V activities lead to updates in one domain, but not in others, since the information is not entered into the FDS. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(66) Data to the MTM is corrupted during manual transfer~~	See (58) above.
~~(67) The state of the MTM is not sufficient, but it is used~~	The following risks are mitigated by data integration creating Data Mining support to analyze the current state of the project and its artifacts: • No clear directive on when the MTM is acceptable. The following risks are mitigated by introducing a formal way of representing the MTM: • Matlab/Simulink Application Designer does not understand the implications of the MTM (lack of training, not present in earlier projects, tight time plan, etc.). Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(68) The state of the SDS is not sufficient, but it is used~~	The following risks are mitigated by data integration creating Data Mining support to analyze the current state of the project and its artifacts: • No clear directive on when the SDS is acceptable. The following risks are mitigated by introducing a formal way of representing the SDS: • CIT Application Engineer does not understand the implications of the SDS (lack of training, not present in earlier projects, tight time plan, etc.). • SDS structured in an informal way (large amount of information in unstructured text, etc.), leading to overconfidence that the SDS is complete and correct. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(69) The state of the FRS is not sufficient, but it is used~~	See (68) above.
~~(70) The state of the MTM is not sufficient, but it is used~~	See (67) above.
~~(71) The SDS, the FRS and the MTM contradict each other~~	See (64) above.
~~(72) The SDS is interpreted differently in different process branches~~	See (65) above.
~~(73) The FRS is interpreted differently in different process branches~~	See (65) above.
~~(81) To create the FIOL takes too long time, due to manual transformation of data. An incomplete version of the FIOL is therefore used~~	The risk is not mitigated by data integration *per se*, but rather centralized into a different form[25].
~~(84) To create the MTM takes too long time, due to manual transformation of data. An incomplete version of the MTM is therefore used~~	See (81) above.
~~(86) Required transformations to M/SAD not constructed prior to launch of tool chain~~	The risk is not mitigated by data integration *per se*, but rather centralized into a different form[26].
~~(87) The M/SAD is created erroneous~~	The risk is not mitigated by data integration *per se*, but rather centralized into a different form[27].
~~(88) Required transformations to FDS not constructed prior to launch of tool chain~~	See (86) above.
~~(89) The FDS is created erroneous~~	See (87) above.

[25] It is centralized into the similar risks associated with (130).
[26] It is centralized into the similar risks associated with (108).
[27] It is transformed into the similar risks associated with (113).

~~(90) Required transformations to MTM not constructed prior to launch of tool chain.~~	See (86) above.
~~(91) Required transformations to Control Application not constructed prior to launch of tool chain.~~	See (86) above.
~~(92) The MTM is created erroneous~~	See (87) above.
~~(93) The Control Application is created erroneous~~	See (87) above.
~~(94) Required transformations to FC not constructed prior to launch of tool chain~~	See (86) above.
~~(95) The FC is created erroneous~~	See (87) above.
~~(96) M/SAD is created incomplete~~	The following risks are mitigated by introducing a specialized GUI (no extended interaction with the IRQA GUI by Non-Expert): • Matlab Application Designer not properly trained in IRQA. • IRQA GUI does not allow for experimentation / give understandable feedback to a non-expert. Based on this I can conclude that this particular casual risk is no longer probable in our context.
~~(97) M/SAD is created erroneous~~	See (96) above.
~~(98) CIT Application is created incomplete~~	See (96) above.
~~(99) CIT Application is created erroneous~~	See (96) above.
~~(100) FDS is created incomplete~~	See (96) above.
~~(101) FDS is created erroneous~~	See (96) above.
(106) Required control not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the required transformation services (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required transformation services does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformation services or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an Artifact require more transformation services. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(107) Parts of services lose safety certification, but are not removed from service catalogue	System Engineer does not identify obsolete transformation services that are left out of software qualification efforts. Requirement to remove obsolete transformation services from service catalogue does not reach the Tool Chain Developers. Tool Chain Developers do not identify all ways that obsolete transformation services are used. Late changes in an Artifact require reintroduction of obsolete transformation services and they are not qualified correctly. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(108) Required transformation not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the required transformations (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement a required transformation does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required transformations or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an Artifact structure require more transformations. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(109) Artifact is created erroneous	System Engineer does not understand the structure of all artifacts (lack of training, tight time plan, etc.). System Engineer does not understand end product environment completely (unknown EMI, etc.). The reliability requirements on the transformations do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the implementation of the transformations (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not understand the structure of all artifacts or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Designers, Engineers and/or Developers will actually use the transformations. Data Integrity is not ensured. No verification of transformations. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(110) Artifact not correctly reproduced.	System Engineer does not understand the structure of all artifacts (lack of training, tight time plan, etc.). The requirements on the scope of the transformations do not reach the Tool Chain Developers. Tool Chain Developers do not understand the implications of the implementation of the transformations (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not understand the structure of all artifacts or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Designers, Engineers and/or Developers will actually use the transformations. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(111) Required control not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the required repository services (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required repository services does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required repository services or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure require more repository services. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(112) Parts of services lose safety certification, but are not removed from service catalogue	System Engineer does not identify obsolete repository services that are left out of software qualification efforts. Requirement to remove obsolete repository services from service catalogue does not reach the Tool Chain Developers. Tool Chain Developers do not identify all ways that obsolete repository services are used. Late changes in an artifact structure require reintroduction of obsolete repository services and they are not qualified correctly. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.

(113) Artifact is created erroneous	System Engineer does not understand the structure of all artifacts (lack of training, tight time plan, etc.). System Engineer does not identify all threats to data integrity. System Engineer does not understand end product environment completely (unknown EMI, etc.). The reliability requirements on DIS do not reach the Tool Chain Developers. Tool Chain Developers do not understand the structure of all artifacts or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not understand the implications of the implementation of the DIS (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Designers, Engineers and/or Developers will actually use the DIS. No verification of the reliability of the DIS. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(114) Artifact not correctly reproduced.	System Engineer does not understand the structure of all artifacts (lack of training, tight time plan, etc.). The requirements on the scope of the data integrity support do not reach the Tool Chain Developers. Tool Chain Developers do not understand the structure of all artifacts or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not understand the implications of the implementation of the DIS (lack of training, not present in earlier projects, tight time plan, etc.). Tool Chain Developers do not identify all scenarios regarding how the Designers, Engineers and/or Developers will actually use the transformations. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(115) Required control not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the traceability services (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required traceability services does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required traceability services or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure require more traceability services. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(116) Parts of services lose safety certification, but are not removed from service catalogue	System Engineer does not identify obsolete traceability services that are left out of software qualification efforts. Requirement to remove obsolete traceability services from service catalogue does not reach the Tool Chain Developers. Tool Chain Developers do not identify all ways that obsolete traceability services are used. Late changes in an artifact structure require reintroduction of obsolete traceability services and they are not qualified correctly. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(117) Artifact is left in erroneous state (incorrect feedback)	System Engineer does not understand the structure of all artifacts in relation to traceability (lack of training, tight time plan, etc.). The requirements on the scope of the traceability granularity do not reach the Tool Chain Developers. Tool Chain Developers do not identify all the different levels of data granularity that have to be supported. Late changes in an artifact structure require more levels of granularity to be supported by traceability. Data Integrity is not ensured. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(118) Required control not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the data mining services (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required data mining services does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required data mining services or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure require more data mining services. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(119) Parts of services lose safety certification, but are not removed from service catalogue	System Engineer does not identify obsolete data mining services that are left out of software qualification efforts. Requirement to remove obsolete data mining services from service catalogue does not reach the Tool Chain Developers. Tool Chain Developers do not identify all ways that obsolete data mining services are used. Late changes in an artifact structure require reintroduction of obsolete data mining services and they are not qualified correctly. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(120) Artifact is left in erroneous state (incorrect feedback)	System Engineer does not identify the full extent to which data needs to be mined and analyzed and some data is not possible to access (lack of training, not present in earlier projects, tight time plan, etc.). The requirements on the access of data for the data mining do not reach the Tool Chain Developers. Tool Chain Developers do not understand the requirements on the access of data for data mining or the implications of that implemented (lack of training, not present in earlier projects, tight time plan, etc.). Data Integrity is not ensured. Late changes in an artifact structure render the implemented data mining obsolete. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(121) Artifact is left in erroneous state (incorrect feedback)	System Engineer does not identify the full extent to which data needs to be mined and analyzed and some required data is not present (lack of training, not present in earlier projects, tight time plan, etc.). The requirements on the scope of the data mining do not reach the Tool Chain Developers. Tool Chain Developers do not understand the scope of the data mining or the implications of that implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure render the implemented data mining obsolete. Insufficient software qualification and data mining fails silently. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(122) Required control not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the process services required (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required process services does not reach the Tool Chain Developers. Tool Chain Developers do not understand the full extent of the required process services or the implications of those implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure, the number of artifacts or process activities require more process services. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(123) Parts of services lose safety certification, but are not removed from service catalogue	System Engineer does not identify obsolete process services that are left out of software qualification efforts. Requirement to remove obsolete process services from service catalogue does not reach the Tool Chain Developers. Tool Chain Developers do not identify all ways that obsolete process services are used. Late changes in an Artifact require reintroduction of obsolete process services and they are not qualified correctly. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(124) Required process logic not constructed prior to launch of tool chain	System Engineer does not understand the full extent of the process logic required (lack of training, not present in earlier projects, tight time plan, etc.). Requirement to implement required process logic does not reach the Tool Chain Developers.

	Tool Chain Developers do not understand the full extent of the required process logic or the implications of that implemented (lack of training, not present in earlier projects, tight time plan, etc.). Late changes in an artifact structure, the number of artifacts or process activities render the process logic obsolete. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(125) Artifact is versioned erroneously and versions are mistaken for each other	Timing constraints are not defined prior to the launch of the tool chain. Requirements regarding timing constraints do not reach the Tool Chain Developers. Implementation of integration aspects do not scale to the size of the artifacts used. Tool Chain Developers do not understand or foresee the timing issues due to the implementation of or interaction between the integration aspects. Timing information in regard to the Tool Chain is not collected and/or analyzed. No feedback on Tool Chain to System Engineer from Domain Experts such as Designers, Engineers or Developers.
(126) To provide feedback takes too long time, due to timing issues with the data mining and necessary feedback is not used	See (125) above.
(127) Notifications are out-of-sync with actual events and wrong version of artifacts are used	See (125) above.
(128) Notifications are out-of-sync with actual events and artifacts are not updated accordingly	See (125) above.
(129) Notifications are not delivered correctly and necessary feedback is not used	See (125) above.
(130) To create the FIOL takes too long time, due to the data integration transformations. An incomplete version of the FIOL is therefore used.	No clear directive on when the FIOL is acceptable. Matlab/Simulink Application Designer does not understand the implications of the FIOL (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of FIOL. Feedback on FIOL completion takes too long time.
(131) To create the CA takes too long time, due to the data integration transformations. An incomplete version of the CA is therefore used.	No clear directive on when the CA is acceptable. CIT Application Engineer does not understand the implications of the CA (lack of training, not present in earlier projects, tight time plan, etc.). (Multiple) Remote calls in PIS or CIS takes too long time. DIS does not scale to current size of CA. Feedback on CA completion takes too long time.

TABLE 10 DIFFERENCES IN CAUSES OF RISKS DUE TO CHANGES IN THE THIRD ITERATION OF THE TOOL CHAIN

9 DISCUSSION

As mentioned in chapter 1, the intention of this technical report is to expand on previous analyses and further define the relation between tool integration and safety. The conclusions regarding this, based on the STPA analysis defined in this technical report, are detailed in the subsequent subchapters. First, however, I present an analysis of the suitability of using STPA for this type of study, since this has not been performed earlier.

9.1 STPA IN THE CONTEXT OF TOOL INTEGRATION - ANALYZED

There were a total of 25 integration weaknesses identified by domain experts and developers using this tool chain on a daily basis. These weaknesses are:

1. Too much FPGA rework. Workflow is based on manually created specification documents.
2. Requirement engineering is poor. There is no traceability to implementation and back.
3. Top level system model(s) are not precise enough and lead to rework later. The implementation does not reflect the top level design.
4. There is no consistent approach across projects on (integration) testing. Hence, this does not allow the creation of a unified process at organization level.
5. Inefficient documentation.
6. The module test (per function) is not documented.
7. The application SW development process is not well described.
8. The application SW development process is not visible.
9. Designers do not work according to a common process.
10. Diversity in tool solutions. There is one common toolbox for the platform, two different toolboxes for the application areas.
11. No possibility to change existent VHDL code from within Matlab Simulink.
12. Requirement engineering is poor. There is no complete list of requirements.
13. Requirement engineering is poor. The textual representation of the existing requirements may lead to different opinions on the meaning.
14. No dedicated top-level system architecture, no system model exists.
15. No SW / HW independent interface descriptions.
16. Manual management of versioning, build, test, deployment, bug reporting/fixes, documentation updates and product updates, etc.
17. Version control is poor.
18. Too much FPGA rework. Workflow regarding realization solutions within the FPGA require several iteration cycles.
19. Too much FPGA rework. Documents are difficult to maintain.
20. No automated procedures exist for re-use of existent, legacy VHDL code into Simulink representations.
21. Inconsistent interface documentation which is difficult to verify.
22. FPGA IO list (in Excel) are error prone and difficult to verify against the FPGA specification.
23. Lack of “global” records of defects leads to the possibilities of repeating the same errors in other projects.
24. Difficult physical access for maintenance is usual. Have to send service personnel to the site or deliver software through internet to local service agents.
25. Ad-hoc testing provides unknown coverage.

Of these, the 20th is contradicted by another project requirement I have identified (that porting code "backwards" through the tool chain should be prohibited so as not to corrupt models with arbitrary changes in the code.); the 15th is not relevant to Safety; and 4, 6, 7, 9, 23, 24 and 25 are not handled, since:

4. Almost no information was given on how many projects run in parallel, how they interrelate (in regard to staff, organization, etc.), which artifacts are carried over, etc. This prohibited analysis (the set of possibilities becomes too complex) except in the case of requirements, which according to the information given are carried over between projects. In the context of this analysis (to analyze the impact of tool integration on safety and vice versa) this omission is most likely not important, since related safety issues belong to a higher hierarchical level.
6. Not enough information was given on this part of the development process, prohibiting analysis. This limitation was understood early and not defined as critical, since large parts of the development process was defined and analyzed.
7. Information was given that parts of the process were not well-defined and that this was one of the reasons for not enough information being communicated regarding these. I believe that the parts that were analyzed were sufficient, but it is doubtful that an STPA analysis could have been performed if the undefined process parts had been required for completeness.
9. See 7 above.
23. See 4 above.
24. See 6 above.
25. Not enough information was given to understand if this is, in fact, an integration issue, prohibiting analysis. This limitation was understood early and not defined as critical, since large parts of the development process was defined and analyzed.

The other integration weaknesses are identified in the STPA analysis either directly through programmatic risks or by the subsequent cause analysis:

1. Identified directly through risks (80), (82) and (83).
2. Identified through causes, for instance that there is no feedback from later development stages on the PRS and TS (i.e. (5)).
3. Identified directly through risks (13) and (14).
5. Identified directly through risks, for instance (46).
8. Identified directly through risks, for instance (64), (65) and (67).
10. Identified directly through risks, for instance (25) and (26).
11. This could lead to safety issues similar to 10 above, but the choice of tools makes this sufficiently unlikely.
12. Identified through causes, for instance that there are different reviewers interspersed throughout the organization with no common feedback on the PRS and TS (i.e. (41)).
13. Identified directly through risks, for instance (41).
14. Identified through causes, for instance that there are different reviewers interspersed throughout the organization with no common feedback on the SDS and MSSM (i.e. (50)).
16. Identified directly through risks, for instance (35).
17. Identified through causes, for instance that the product manager may not be informed about new version of requirements to the PRS (i.e. (35)).
18. Identified directly through risk (83).
19. Identified directly through risks (80) and (82).
21. Identified directly through risk (64).
22. Identified directly through risk (64).

Most of the unhandled weaknesses, and associated risks, were not identified because I did not have enough expertise in regard to the details of the case study (I did for instance not know the scope of the information that our industrial partner transferred between projects). Other unhandled weaknesses were not handled due to process parts for which information was not readily available. However, in hindsight these omissions seem to be bounded, so they should not affect the validity of the rest of the analysis.

Two positive outcomes of applying STPA were highlighted during this case study; first, that STPA allowed the weaknesses identified by domain experts and developers to be systematized into risks or causes; secondly that more risks and causes could be identified and detailed by the guidance given by STPA, than through unsystematic feedback based on expert knowledge.

The following conclusions can be drawn:

- STPA was designed to provide guidance during hazard analysis. Within the context of tool integration it can manage this, both in regard to identifying and categorizing risks and causes.
- STPA is not a silver bullet. To apply STPA still requires a lot of expertise in the area to be analyzed, since it is otherwise easy to omit important parts of the process models.

9.2 SYSTEM-WIDE PROPERTIES

When analyzing the results the lack of some properties is found as causes to many of the risks.

1. Lack of *Data Integrity* can cause artifacts to be reproduced incorrectly (see for instance (13)), incompletely captured (see for instance (18)) and corrupted during manual transfer (see for instance (61)).
2. Lack of *Traceability for Completeness and Consistency* can cause artifacts to be reproduced incorrectly (see for instance (13)) and incompletely captured (see for instance (18)).
3. Lack of *Automated Transformations of Data* can cause artifacts to be reproduced incorrectly (see for instance (13)).
4. Lack of *Formally Defined Data Semantics* can cause artifacts to be reproduced incorrectly (see for instance (17)) and incompletely captured (see for instance (22)).
5. Lack of *Possibility to Automate Tool Usage* can cause artifacts to be created incomplete (see of instance (27)) and erroneous (see for instance (28)).
6. Lack of *Data Mining* can cause artifacts to be used although their state is insufficient (see for instance (46)).
7. Lack of *Process Notifications and Process Control* can cause artifacts to be corrupted during manual transfer (see for instance (61)), artifacts to be used although their state is insufficient (see for instance (46)), artifacts to contradict each other (see for instance (52)) and artifacts to be interpreted differently in different process branches (see for instance (65)).
8. Lack of *Possibility to Create Customized GUIs* can cause artifacts to be created incomplete (see of instance (96)) and erroneous (see for instance (97)).

A number of statements can be made regarding these properties:

- The effectiveness of these properties in doing their part to ensure safety in the end product is directly related to how large part of the development environment they are supported in.

    - The properties require support throughout a substantial part of the development environment to have a noticeable impact. There were 85 risks found when analyzing the OTC. When the SITC was analyzed, only 7 risks were found to be mitigated by ensuring these properties at different parts of the SITC. At the same time 14 new risks (out of 20 added) were added due to the tool integration mechanisms introduced to ensure these properties (leading to a net increase of risks). When the TITC was analyzed, the numbers were 51 and 26 respectively (leading to a net reduction of risks).
    - A large part of the reason for the net reduction of risks in TITC was the possibility to *centralize* the support of these properties. However, out of the 26 new risks added in the TITC, 24 were found in the centralized support of these properties. Any multiple setups of support of these properties would generate a similar amount of risks (multiple setups can easily be required if the development environment is divided into several *islands of automation* [4]).
    - Introducing support for these properties across the **whole** development environment is likely to be costly, or even impossible. An example of this would be to try to introduce fully automatic data transformations between requirements and detailed design. Capturing enough information to enable this has not been deemed likely to succeed in the case study at hand, since it would require either specialists (for instance the Matlab Application Designer) to engage in the development process at a much higher level of abstraction than usual, or high level experts (for instance the System Architect) to study the required domains in detail. This has lead to parts of the SITC and TITC retaining several of the risks mitigated at other parts. An example of this is how (9) is not mitigated, but (13) is.
- The introduction of tool integration mechanisms to support these properties is not a silver bullet for ensuring safety.
    - While some causes were mitigated, some were only transformed into new causes to new risks. An example of this is how the lack of understanding on part of the Matlab Application Designer can cause (13) is transformed into how the lack of understanding on part of Tool Chain Developers can cause (109). In fact, one can argue that these new causes are more severe than the original ones, since they are one step removed from the fallout of the risks.
        - First, if the cause is not identified and results in a hazardous event, this event is more unlikely to be handled. In the example above it is more likely that the Matlab Application Designer will notice that the SDS is not completely reproduced, than the Tool Chain Developers who may not even be part of the development process.
        - Secondly, even if the cause is identified, it is more difficult to mitigate. In the example above (109) requires an organization based on the structured propagation of specific safety requirements from higher levels of the organization to the Tool Chain Developers, while the equivalent requirements for (13) is straight-forward to organize within the development process.
    - To mitigate one risk, a combination of causes must often be mitigated by a combination of these properties. An example of this is how (13) requires Data Integrity, Traceability for Completeness and Consistency and Automated Transformations to be fully mitigated.

To summarize these findings one can state that a centralized support of all of these properties in as much as possible of the development environment is likely to yield a “safer” tool chain, but

that a higher cost will have to be paid at higher levels of organization. Which choice, out of adding support or spending the effort on other control mechanisms (for instance training operators), is the best will be defined by:

- The size of the development environment, since a larger environment will mean a larger number of risks mitigated by each centralized service.
- The number of contexts towards which the centralized services can be verified, since a centralized service imply generalization and sharing of the verification effort (for instance by all companies using the same integration framework).
- The effort required to ensure the communication of relevant safety requirements between the experts of day-to-day operations and tool chain developers.

9.3 SOFTWARE QUALIFICATION

What are then the implications of the findings from subchapter 9.2 on a software qualification effort?

One can start with looking at the risks left in the middle part of the tool chain at hand, i.e. (14), (16), (19), (20), (51), (53), (60), (74), (75), (76), (77), (78), (79), (80), (83), (85), (102), (103), (104), (105), (130) and (131). Based on these one can make a number of further statements:

- The high levels of abstraction at the beginning and end of the tool chain contaminate the middle part with risks, but only at the start and end points of the middle part. Examples of this is (14), where fully automated data transformations are not available, and (83), where the simulation results still have to be interpreted by the FPGA Application Engineer. As mentioned earlier, moving between abstraction levels is associated with risks if done manually, but usually very costly to automate.
- In the TITC, the causes originating in software are no longer likely for most of these risks (namely (14), (16), (19), (20), (51), (53), (60), (74), (75), (76), (77), (78), (79), (80), (83) and (85)). These risks still require software qualification of the centralized services supporting the properties described in subchapter 9.2, but are otherwise rather dependent on efforts at higher levels of organization. Examples of the increased dependency on higher levels of organization to ensure safety is noticeable throughout the TITC; just adding process notifications doesn´t ensure that communication will occur; just adding data mining will not ensure that data quality is ensured during times of increased pressure on project time constraints; etc.
- Some risks remain unaffected by support for these properties, i.e. (102), (103), (104), (105), (130) and (131). This is because they can be caused by the platform and relate to something as complicated to analyze as time. One can however envision ensuring this through a global clock and efforts at higher levels of organization.

Based on this I can deduce yet another property that when lacking can lead to safety violations:

9. Lack of *Coherent Time Information* (production time synchronized to a global clock, etc.) in development artifacts can cause incomplete versions of artifacts to be used.

The actual impact on software qualification then depends on how this effort is approached. In the case of a primarily prescriptive safety standard this, if relevant, is described in detail and the findings in this report will have little or no impact. In the case of a primarily descriptive safety standard the impact can take many forms and I base the following discussion on the newly released ISO 26262 standard. The approaches that can be considered according to ISO 26262 would be:

- To qualify all tool integration mechanisms separately, as separate software tools. This would essentially follow the simple “more software, more software qualification” logic [6].
- To apply the guidelines set down by the safety standard in one allowed, generic way for all tool integration mechanisms so that one achieves a beneficial synergy effect. In ISO 26262 one of the ways of qualifying software tools, which is deemed applicable when developing the most safety-critical systems, is to identify that they have been developed according to a safety standard. This safety standard can be ISO 26262 itself. ISO 26262 describes the procedure of developing a Safety Element out of Context (SEooC), i.e. a safety-related element that is not developed to be used in any, specific item (it is developed to be reusable). The development of a SEooC involves making assumptions on the existing conditions of the corresponding phase in the product development and then to check the validity of these assumptions in the context of the actual item after integration of the SEooC. I argue that a software tool that is developed to be reusable in several different tool chains **and** always qualified in the context of tool chains fit this description well. Given this approach software tools and tool integration mechanisms can be developed and pre-qualified by **tool vendors** based on the assumption that other parts of the development environment will provide certain support. In this way the required Safety Integrity Level of specific tools or tool integration mechanisms, and thereby the effort for the pre-qualification, qualification and development, can be lowered.

To summarize, blindly increasing the amount of tool integration between software tools will only serve to increase the required software qualification effort; strategically targeting a few system-wide properties with relevant software and efforts at a high level of organization can however serve to lower the required software qualification effort.

Once again, if this is the best way of approaching software qualification will depend on the size of the development environment, the possible contexts for verification and the effort required at higher levels of organization.

10 SUMMARY AND CONCLUSIONS

The goal of this report was to detail a HAT analysis of one of the iFEST case studies and perform a subsequent analysis of the impact of safety on tool integration (and vice versa). The HAT of choice became STPA, since it has been designed to provide guidance, handle different domains efficiently, be possible to use before a system has been built and be used in the context of software.

However, STPA had not been used in the context of tool integration before, which meant that I first had to customize STPA to this context. This meant that I had to perform two things; first I had to translate the basic possibilities for inadequate control for the hazard that tool integration contributes or leads to an unsafe product being produced into general types of risks; secondly, I had to identify the equivalent of the control loop entities used in a STPA (i.e. the controller, the controlled process, the actuators and the sensors) in this context. Subchapter 3.1 lists the translation and a summary of the general types of risks in our context. Chapter 8 details the identification of the control loop entities important to STPA. Both of these chapters build on the notion that a "safe" tool integration mechanism is one that does not by itself contribute to the accumulation of defects that can violate safety constraints in the end product.

Based on this customization three different versions of a tool chain were analyzed and the result compared to the integration weaknesses identified by domain experts and developers using the original tool chain on a daily basis. A net result of 85, 98 and 73 risks was identified, in comparison to 25 integration weaknesses. Furthermore, out of the 16 issues that were identified by both methods, 12 could be identified as risks and 4 as causes by STPA. In fact, 1 of these issues could immediately be disregarded as highly unlikely due to STPA. Of the 9 other integration weaknesses, 7 were found to be applicable to our context and not identified during the STPA analysis due to missing information (either information unknown to me or not available at all). The following conclusions could be drawn:

- STPA was designed to provide guidance during hazard analysis. Within the context of tool integration it can manage this, both in regard to identifying and categorizing risks and causes.
- STPA is not a silver bullet. To apply STPA still requires a lot of expertise in the area to be analyzed, since it is otherwise easy to omit important parts of the process models.

In regard to the relationship between safety and tool integration 9 properties were identified, properties that need to be supported correctly to avoid hazards in our context. These are detailed in subchapter 9.2 and 9.3. In regard to these properties the following conclusions could be drawn:

- They require support throughout a noticeable part of a development environment to have an impact and derive much of their impact from the possibility to centralize them.
- They interrelate, so that several of them often need to be handled to mitigate one type of risk.
- Introducing support for them across a whole development environment is likely to be costly, or even impossible.
- These properties are not a silver bullet. Introducing support for them will mitigate some risks, but also create others at higher levels of organization.

Based on the discussion regarding these properties I could also draw a number of conclusions regarding how to handle them and enable a more efficient qualification effort based on them:

- The effectiveness of adding support for these properties will depend on the size and of the development environment, the possibility to centralize supporting services, the number of contexts towards which the supporting services can be verified and the effort required for ensuring the associated added requirements at higher levels of organization.
- Tool and tool chain developers could develop and pre-qualify their software based on the assumption that some or all of these properties are supported by the final development environment. The final qualification effort can thus be lowered or at least distributed more efficiently.

11 BIBLIOGRAPHY

1. *Tool Integration, from Tool to Tool Chain with ISO26262.* **Asplund, Fredrik, et al.**

2. **Leveson, Nancy G.** *Engineering a Safer World (Draft).* s.l. : MIT Press, 2011. Boston.

3. *Controversy Corner: A new research agenda for tool integration, J. Syst. Softw.* **Wicks, M. N. and Dewar, R. G.** 9, New York, NY, USA : Elsevier Science Inc., 2007, Vol. 80. ISSN: 0164-1212.

4. *Tool integration beyond Wasserman, Advanced Information Systems Engineering Workshops: CAiSE 2011 International Workshops.* **Asplund, Fredrik, et al.** London, UK : s.n., 2011.

5. *Certification Specifications for Very Light Rotorcraft, CS-VLR.* s.l. : European Aviation Safety Agency, 2008.

6. **International Organization for Standardization.** *ISO 26262 - Road vehicles - Functional safety.* s.l. : International Organization for Standardization, 2011.

7. **European Committee for Electrotechnical Standardization.** *BS EN 50128:2001 - Railway applications — Communications, signalling and processing systems - Software for railway control and protection systems.* s.l. : European Committee for Electrotechnical Standardization, 2001.

8. **Special Committee 167 of RTCA, Inc.** *DO-178B, Software Considerations in Airborne Systems and Equipment Certification.* s.l. : RTCA, Inc, 1992.

9. **Hamann, Reinhold, et al.** *ISO 26262 Release Just Ahead - Remaining Problems and Proposals for Solutions.* s.l. : SAE, 2011.

10. **Storey, Neil.** *Safety-Critical Computer Systems.* Harlow : Addison Wesley Longman, 1996. ISBN 0-201-42787-7.

11. *Journal of Hazardous Materials, Hazard and operability (HAZOP) analysis. A literature review.* **Dunjó, Jordi, et al.** 1-3, 2009 : Elsevier B.V., 2010, Vol. 173. ISSN: 0304-3894.

12. **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.* s.l. : British Standards Institution on ERC Specs and Standards, 2003.