



Norwegian University of
Science and Technology

Prototyping a digital support tool for an agile implementation of STPA

Niklas Ludvigsen

Master of Science in Computer Science

Submission date: July 2018

Supervisor: Jingyue Li, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Systems Theoretic Process Analysis (STPA) is a powerful new hazard analysis method for safety engineering. This method aims to support a world where systems are more complex as a result of software. But this method might not support an environment where there is both complexity and incremental changes, such as an agile software development environment. In such an environment where there is both complexity and change, it might be preferable or needed to have digital support tools for assisting the method. This thesis aims to explore the possibility of such a tool and what functionality it could provide to an agile software development environment.

In this thesis this was explored by generating a full set of suggestions and requirements based on findings from literature and experience. These requirements were then used to implement a prototype that aimed to test these requirements.

The implemented systems functionality was then compared to other existing implementations of STPA software assistance.

The prototype was found to be currently lacking, but having a promising future if further implemented.

Sammendrag

Systems Theoretic Process Analysis (STPA) er en kraftig ny risikoanalysemetode for “safety engineering” . Denne metoden tar sikte på å støtte en verden der systemer er mer komplekse som følge av programvare. Men selv denne metoden kan kanskje ikke støtte et miljø der det er både kompleksitet og inkrementelle endringer, for eksempel et “agile software development environment”. I et slikt miljø der det er både kompleksitet og forandring, kan det være å foretrekke eller behøve å ha digitale støtteverktøy for å bistå metoden. Denne oppgaven tar sikte på å undersøke muligheten for et slikt verktøy og hvilken funksjonalitet det kan gi et “agile software development environment”.

I denne oppgaven ble dette utforsket ved å generere et komplett sett med forslag og krav for et slikt system, basert på funn fra litteratur og erfaring. Disse kravene ble så brukt til å implementere en prototype som skulle teste disse kravene.

Den implementerte systemfunksjonaliteten ble da sammenlignet med andre eksisterende implementeringer av STPA-programvarehjelp.

Prototypen har for øyeblikket mangler, men å kunne ha en lovende fremtid, hvis den ble videreutviklet.

Preface

This thesis was written as a final delivery for a Master of Technology degree in Computer Science. With this delivery the author graduates from the Department of Computer Science at University of Science and Technology (NTNU).

I would like to thank my supervisor Associate Professor Jingyue Li (Bill) for his guidance throughout this project.

Sandefjord, July 2018

Contents

Abstract	i
Sammendrag	ii
Preface	iii
1 Introduction	1
1.1 Motivation and Background	1
1.2 Research Questions	2
1.3 Previous Work	2
1.4 Contributions	2
1.5 Structure of the thesis	3
2 State of the Art	5
2.1 System-Theoretic Accident Model and Processes (STAMP)	5
2.2 System-Theoretic Process Analysis (STPA)	6
2.2.1 Fundamentals	6
2.2.2 Step 1	8
2.2.3 Step 2	10
2.3 Extending and automating STPA	11
2.3.1 Context Variables	11
2.3.2 Traceability for STPA	12
2.3.3 Automating and Simplifying STPA	12
2.3.4 Agile Software Development Using STPA	13
2.3.5 System-Theoretic Process Analysis - Security(STPA-Sec)	14
2.4 Current STPA Tools	14
2.4.1 XSTAMPP using the A-STPA plugin	14
2.4.2 SafetyHAT Project	16

2.4.3	Sahra	16
2.4.4	RM Studio	16
2.4.5	STAMP Workbench	16
2.4.6	SpecTRM	17
2.4.7	An STPA Tool	17
2.5	Incremental Software Development	17
3	Research Method	18
3.1	Motivation	18
3.2	Choice of Research Questions	18
3.3	Method	19
3.3.1	Interviews	19
3.3.2	Software Prototype	19
3.4	Limitations	20
4	What functionality could a software system provide the STPA method?	21
4.1	Establishing Fundamentals	21
4.2	STPA Step 1	22
4.3	Control Structure	24
4.4	Agile software development team	25
4.5	Hierarchical Structure	26
5	Results of Research Question 1	27
5.1	Architectural Requirements	27
5.1.1	The Goals of the System	27
5.1.2	Functional Requirements	28
5.1.3	Quality Requirements	33
6	Results of Research Question 2	37
6.1	Back End	37
6.1.1	Framework	37
6.1.2	Database	38
6.2	Front End	41
6.2.1	Templates	41

6.3	Requirements Implementation	41
6.3.1	Storage and User Profile	41
6.3.2	Authentication	41
6.3.3	STPA fundamentals	42
6.3.4	Extending STPA	42
6.3.5	Version Control	43
6.3.6	Traceability	44
6.4	Implementation Result	44
6.4.1	FR 4.3-4.5	45
6.4.2	FR 5.2	45
6.4.3	FR 6.2	45
6.4.4	FR 7	46
7	Comparison of current software tools	47
7.1	Compared versions	47
7.1.1	STPA Prototype	47
7.1.2	XSTAMPP	48
7.1.3	Stamp Workbench	48
7.1.4	SafetyHAT	48
7.1.5	An STPA Tool	48
7.1.6	SAHRA	48
7.2	Establishing Fundamentals Suggestions	48
7.2.1	List management	49
7.2.2	Traceability	49
7.2.3	Guide	50
7.3	STPA Step 1	50
7.3.1	HCA Table	50
7.3.2	Context Table	50
7.3.3	Linking HCA to Hazards	50
7.3.4	HCA to Hazard	51
7.3.5	Logical Simplification	51
7.3.6	Continuous Process Model Variables	51
7.3.7	Hazard Rules	51

7.4	Control Structure	51
7.4.1	Fundamentals from Control Structure	52
7.4.2	Importing or Exporting Control Structure	52
7.5	Agile software development team	53
7.5.1	Re-evaluation suggestion	53
7.5.2	Version control	53
7.5.3	Team management	53
7.6	Hierarchical Structure	54
7.6.1	Communication	54
7.6.2	Abstraction views	54
8	Conclusion and Future Work	55
8.1	Conclusion	55
8.1.1	Conclusion	55
8.2	Future Work	55
A	Setup	60
A.1	Start the system	60

List of Figures

2.1	Generic example of a hierarchical control structure, from Leveson [11]	7
2.2	A classification of control flaws leading to hazards, from Leveson [11]	10
2.3	The XSTAMPP architecture from their homepage [23]	15
6.1	ER diagram of the databases	40
6.2	A view of updating a PMV and its associated PMV values	42
6.3	A view of the context table for the train example	43
6.4	A view of the current Git commits	44

List of Tables

2.1	A set of examples hazards for the train case, from Thomas [26]	9
2.2	A example table for identifying hazardous control actions, from Thomas [26]	9
2.3	Decomposing context into variable and values, from Thomas [26]	11
2.4	A example context table for identifying hazardous control actions, from Thomas [26]	12
2.5	STPA to STPA-Sec	14
4.1	Identified possible support functionality, for establishing fundamentals	22
4.2	Identified possible support functionality, for STPA step 1	24
4.3	Identified possible support functionality, for control structure	25
4.4	Identified possible support functionality, for change management	26
4.5	Identified possible support functionality, for hierarchical STPA analysis	26
5.1	Database Requirements	28
5.2	User Profile Requirements	28
5.3	Authentication Requirements	29
5.4	STPA Step 1 Requirements	29
5.5	Context Table Requirements	30
5.6	Automating and Simplifying Requirements	30
5.7	Agile Software Development Requirements	31
5.8	Traceability Requirements	32
5.9	STPA-Sec Requirements	32
5.10	Quality Requirement U1	33
5.11	Quality Requirement U2	34
5.12	Quality Requirement U3	34
5.13	Quality Requirement M1	35

5.14	Quality Requirement M2	35
5.15	Quality Requirement I1	36
5.16	Quality Requirement I2	36
6.1	Functional requirements covered by the implementation	45
7.1	Establishing fundamentals tools overview	49
7.2	STPA step 1 support	51
7.3	Control structure support	52
7.4	Support for agile software development teams	53
7.5	Support hierarchical structure	54

List of source codes

6.1	Defining the hazards blueprint and index page	38
6.2	Registering the hazards blueprint	38
6.3	The Project table in the user database schema	39
6.4	The start of the ProjectDB class, that takes a project id and provides a ORM for that project database	39
6.5	Using GitPython create a new project, and adding an initial commit	44

Chapter 1

Introduction

Systems Theoretic Process Analysis (STPA) is a powerful new hazard analysis tool for safety engineering. While this approach aims to support a world where systems are more complex because of software, the method might be lacking sufficient software support tools for agile software development. In an agile software development environment there would be a need for the approach to support functionality such as providing proper information to the analyst, when there are agile changes to the system being analysed.

1.1 Motivation and Background

STPA analysis is a method that uses accident, its linked hazard, a control structure and control actions to identify hazardous control actions (HCA) that the system might perform. The analysis then uses these HCAs to identify what could cause them and what constraints need to be defined to prevent them. As this analysis is a newer safety hazard analysis, there are steps of this method that might be argued to be ad-hoc [26]. To make the method more formal John Thomas presented the Systematic Method for performing STPA analysis [26]. This method utilizes process model variables (PMV) and their values, to make analysis more formal and systematic. These PMVs are variables that define a components model for the surrounding world. For example, a train door controller needs to have a model of the surrounding state of the train, to know when it is appropriate to issue an open door control action.

The introduction of STPA could be an advantageous tool for agile development teams, but the implementation of performing this analysis could be a resource heavy process not viable for modern agile development team processes [12]. To help this restriction on the analysis

Thomas through several of his papers suggests possible ways to automate and simplify several of the steps in the STPA process when using his Systematic Method [25] [26].

1.2 Research Questions

The goal of the thesis will be to explore the possibility of creating an application that could support the STPA step 1 approach in an agile development environment. It will focus on step 1 as this step helps identify possible hazardous actions, while the analysis why might need to be different in a software development environment. This will be done by answering three research questions that seeks to answer how a STPA tool would perform in an agile software development environment.

RQ1 What requirements would be needed for an STPA application to support an agile development environment?

RQ2 How could the requirements presented in RQ1 be implemented?

RQ3 How would such a tool compare to current solutions for STPA analysis?

To help answer these questions, there will be developed a prototype application based on collected requirements, including tools for supporting agile software development teams. This will then be compared to other applications and tools to explore the effect of these additions.

1.3 Previous Work

In the authors specialization project [12] for the course “Computer Science, Specialization Project (TDT4501)” [13], the possibility of applying STPA to an agile software development environment was explored. It was found that with current solutions that this might not be a good fit, but could maybe be possible with proper software support tools.

1.4 Contributions

The contributions of this thesis are:

- A literature review of the current state of STPA, STAMP and associated software support tools
- A list of suggestions for what software tools could provide STPA analysis in an agile software environment
- A full set of requirements for a system assisting STPA analysis
- A prototype that implements functionality to support agile software development in STPA analysis
- A comparison of current software assistance tools, for performing STPA analysis

1.5 Structure of the thesis

This thesis is structured around the research question, and the chapters are as follows:

- **Chapter 1 - Introduction** introduces the thesis and lays the foundation for the following chapters.
- **Chapter 2 - State of the Art** presents the literature review for the current state of STAMP, STPA, agile development and the current tools assisting STPA.
- **Chapter 3 - Research Method** presents and discuss the research motivations, research questions and research methods of this thesis.
- **Chapter 4 - What functionality could a software system provide the STPA method?** Using the findings of the literature review to present suggestions for how software could assist STPA analysis in an agile software development environment.
- **Chapter 5 - Results of Research Question 1** answers RQ1 by providing a full set of requirements based on the findings in previous chapters.
- **Chapter 6 - Results of Research Question 2** answers RQ2 by presenting the implementation of a STPA prototype accompanying this thesis.
- **Chapter 7 - Comparison of current software tools** answers RQ3 by providing comparison of all accessible current solution for software assisted STPA analysis, including the prototype accompanying this thesis.

- **Chapter 8 - Conclusion and Future Work** concludes this thesis by presenting what was achieved in this thesis, and presents possibilities for future work.

Chapter 2

State of the Art

In this chapter we will perform a literature review of the current state of STAMP, STPA and agile software development. First in section 2.1 and section 2.2 we will look at STPA analysis, and the model STPA is based on, STAMP. Then in section 2.3 and section 2.4 we will look at the tools and techniques that expand STPA analysis. Lastly in section 2.5 we will look at the agile software development approach.

2.1 System-Theoretic Accident Model and Processes (STAMP)

System-Theoretic Accident Model and Processes (STAMP) is an accident model for safety engineering, first introduced by Nancy Leveson in 2002 [8][9]. This model aims to achieve a better fit to the modern complex systems, than the more classic engineering approaches. She proposes that the environment has drastically changed since the more classic engineering approaches was suggested, and has not been able to keep pace. This change is contributed to software, and she argues that this has revolutionized engineering [10].

Three basic constructs underlie the accident model: safety constraints, hierarchical safety control structures, and process models. In STAMP the most basic concept is a constraint, where these safety constraints work in hierarchical safety control structures, where it is the job of higher levels to enforce the constraints on lower levels. As such events leading to loss occur when safety constraints are not successfully enforced by higher levels [11]. A generic example of such a hierarchical structure is shown in Figure 2.1. For this hierarchical structure to work there is a need for effective communication between the different levels of the structure, both a downward reference channel providing the information necessary to impose safety

constraints, and an upward measuring channel about how the constraints are being satisfied. The communication downward from higher levels comes in the form of *control actions*, that activate *actuators* for the lower levels. Leveson [11] gives reasons for why a level may give inadequate control as missing constraints, inadequate safety control commands, commands that were not executed correctly at a lower level, of inadequately communicated or processed feedback about constraint enforcement.

The third basic concept of STAMP, Process Models, are about the model of the process that a higher level must have to keep track of what state a lower level is in. This process model could be a mental model maintained by a human controller, or it could be a digital storage, but it still needs to contain the same type of information. It needs the information of the required relationship among the system variable, the current state, and the ways the process can change states [11].

2.2 System-Theoretic Process Analysis (STPA)

While STAMP provide a new model for safety engineering, STPA and CAST are the use of this model in practise. STPA try to answer the question “How do we find inadequate control in a design?” [25]. This is done through two main steps, identifying the potential for inadequate control of the system that could lead to a hazardous state, and determine how each potentially hazardous control action could occur.

2.2.1 Fundamentals

Before starting the STPA process, the method requires the user to define top level accidents, hazards, safety requirements and constraints and the safety control structure. Leveson [11] defines these in the context of the approach as such:

Accident: An undesired or unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission loss, etc.

Hazard: A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss).

An accident does not necessarily need to be human loss, but could be any loss that the stakeholders deem unacceptable. This could involve states that compromise the goals of the system,

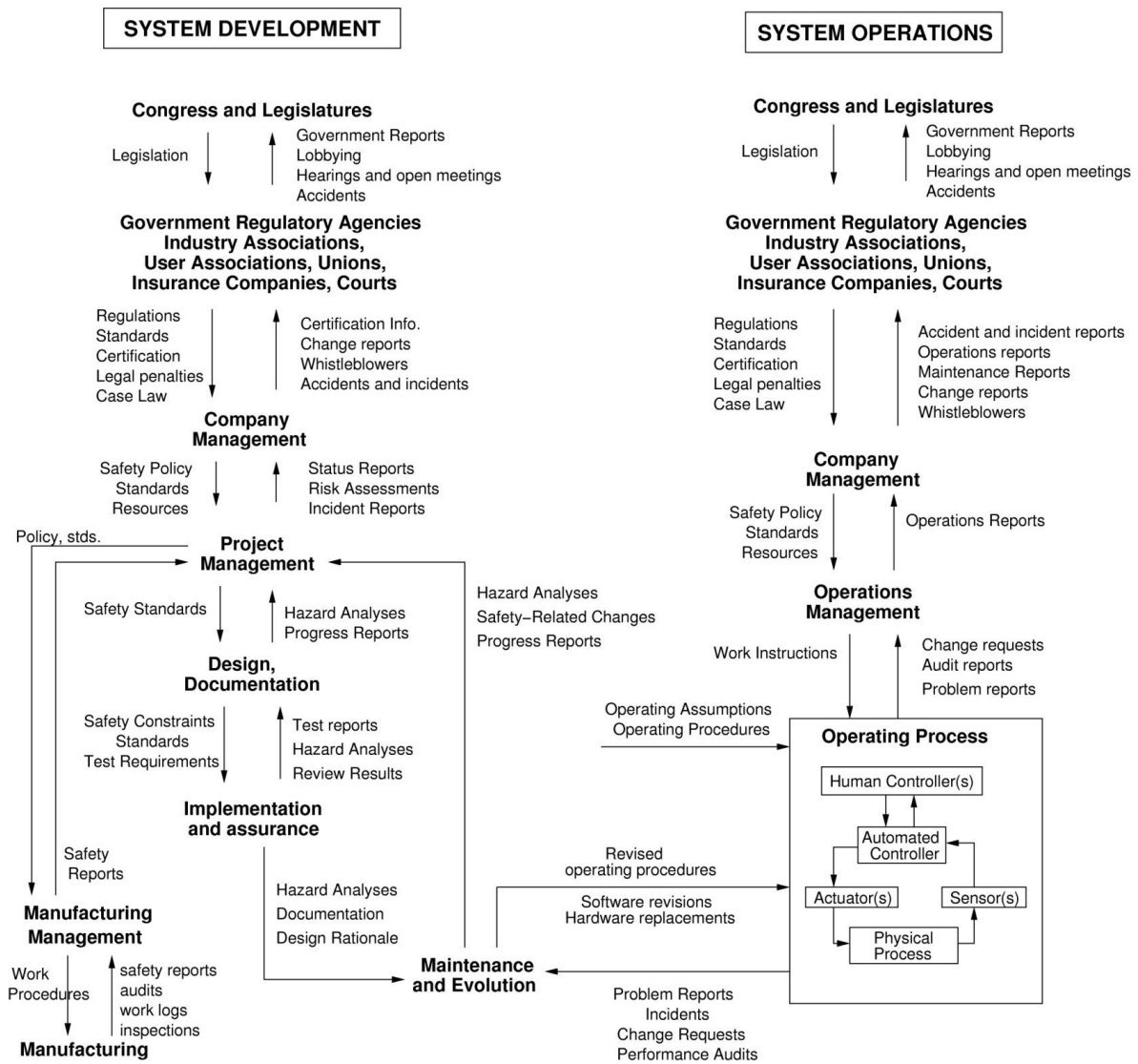


Figure 2.1: Generic example of a hierarchical control structure, from Leveson [11]

such as if hindering the user from using unauthorized resources is a mission of the system, failing this would be an accident. This case would not directly cause a loss for any one person but as the system is not able to fulfil its mission, and as such may be unacceptable by the stakeholders of the system. Leveson also suggests defining levels of loss for systems, that would help to prioritise certain accidents, when there is a need of trade-off among goals in the design process.

Leveson argues that by limiting the definition of hazard to a state the system never should be in, the designer achieves more freedom and ability to design hazards out of the system. But as her definition say, both state and conditions are possible, as long as they could lead to an accident.

After the top-level hazards and accidents has been identified, Leveson suggests providing the system and components with safety requirements and safety design constraints. To help identify these top-level constraints and requirements, the designer may use the previously identified hazards to identify what constraints and requirements are necessary to prevent these hazards from happening. These are to later be refined through several iterations throughout the STPA process, as they are only top-level constraints identified before the main STPA steps, and as such are highly unlikely complete.

Last step of establishing the fundamentals is to draw the control structure of the system. This control structure should provide the user with information over the different components of the system, and may provide some information over the control actions and feedback in the system. Leveson states that there is no one correct safety structure, and the structure should be practical and effective for the designer and the system.

2.2.2 Step 1

The first main step of STPA aim to identify more unsafe states that is not identified in the top-level hazards that the fundamental step aims to discover. This is done by looking at each control action available, and evaluate how this control action could lead to a hazardous state. Leveson provides four activations that a control action may provide, that should be evaluated for leading to a hazardous state.

1. A control action required for safety is not provided or is not followed.
2. An unsafe control action is provided that leads to a hazard.

Table 2.1: A set of examples hazards for the train case, from Thomas [26]

Hazard	Description
H-1	Doors close on a person in the doorway
H-2	Doors open when the train is moving or not in a station
H-3	Passengers/staff are unable to exit during an emergency

Table 2.2: A example table for identifying hazardous control actions, from Thomas [26]

ID	Control action	Control action hazardous			
		Not providing causes hazard	Providing causes hazard	Wrong timing / order causes hazard	Stopped too soon or applied too long
CA1	Provides door open command	Doors not commanded open once train stops at a platform [not hazardous] Doors not commanded open for emergency evacuation [see H-3] Doors not commanded open after closing while a person or obstacle is in the doorway [see H-1]	Doors commanded open while train is in motion [see H-2] Doors commanded open while train is not aligned at a platform [see H-2]	Doors commanded open before train has stopped or after it started moving (same as “while train is in motion”) [see H-2] Doors commanded open late, after train has stopped [not hazardous] Doors commanded open late after emergency situation [see H-3]	Door open stopped too soon during normal stop [not hazardous] Door open stopped too soon during emergency stop [see H-3]
CA2	Provides door close command	Doors not commanded closed or re-closed before moving [see H-2]	Doors commanded closed while person or object is in the doorway [see H-1] Doors commanded closed during an emergency evacuation [see H-3]	Doors commanded closed too early, before passengers finish entering/exiting [see H-1] Doors commanded closed too late, after train starts moving [see H-2]	Door close stopped too soon, not completely closed [see H-2]

3. A potentially safe control action is provided too late, too early or out of sequence.
4. A safe control action is stopped too soon or applied too long (for a continuous or non-discrete control action).

A visual representation of how these control actions are related to each other can be seen in Figure 2.2, and a table for identifying hazardous control actions can be seen in table 2.2. In the table you can see control actions argued for why they may be hazardous in certain context, and linked to what hazard they may cause. A table introducing the linked hazards may be seen in table 2.1. The used case for this example uses the situation of a train door controller, that should ensure that the doors are opened only when it is safe.

Later articles after Leveson [11], suggest that this identification process should be linked to the top-level hazards identified when establishing the fundamentals of the system [26] [7]. As suggested this may be either be done by linking each hazardous control action to one or more system hazards, or by looking at each hazard and identifying what control actions could lead to it.

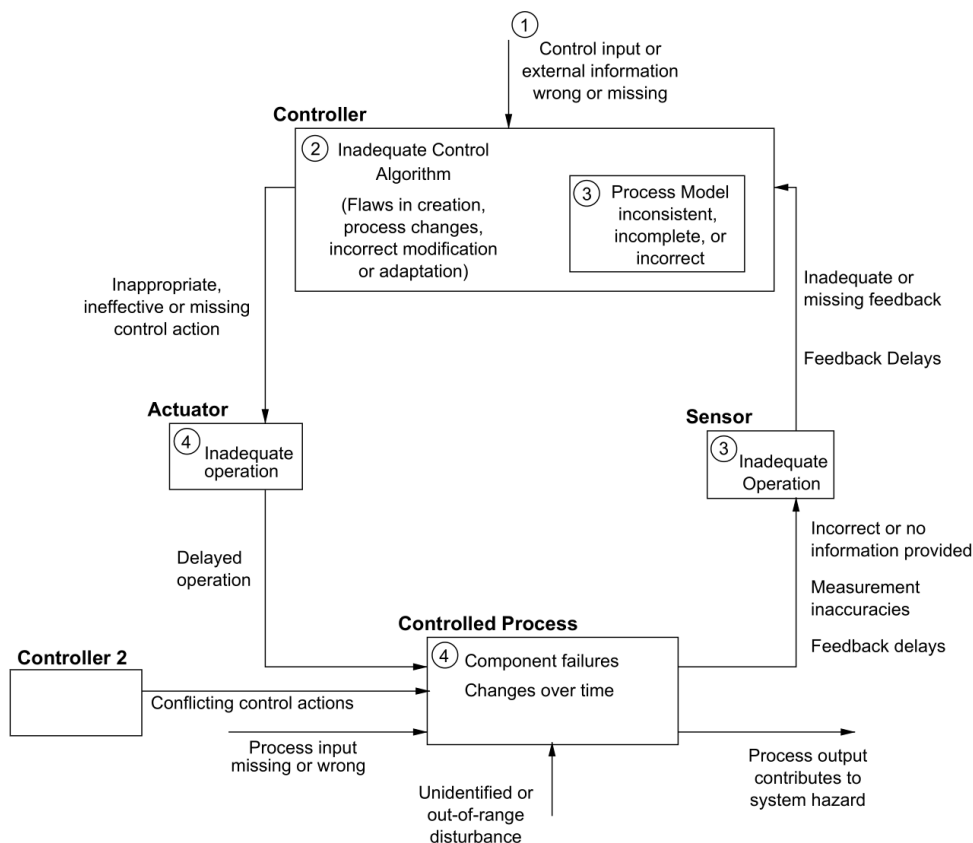


Figure 2.2: A classification of control flaws leading to hazards, from Leveson [11]

When these hazardous control actions are identified, it is possible to use these to identify more safety constraints that are needed for the system. Each hazardous control action may be evaluated, and new constraints that prevent these may be provided.

2.2.3 Step 2

Step 2 of the STPA process is to provide how the potentially hazardous control actions identified in step 1 might happen. This is not a necessary step for all systems as for some systems the constraints and requirements identified in step 1 may be sufficient. While some systems do not need step 2, for others step 1 does not provide sufficient analysis to provide safety to the system. While step 1 help identify how to control actions are used correctly, but safety constraints are still broken. This step follows the more classic hazard analysis in identifying, what and how the system could fail such that a hazard happens.

To analyse why a hazard could happen, the control loop for each hazardous control action are evaluated. The parts of this control loop are examined to determine if they could cause or contribute to the hazard. Using these identified causes or contribution may then be used as a

Table 2.3: Decomposing context into variable and values, from Thomas [26]

Context variables	Context values
Train motion	Stopped
	Moving
Train location	At platform
	Not at platform

tool to identify where there is a need for mitigation and prevention techniques.

2.3 Extending and automating STPA

In 2013 John Thomas wrote a thesis suggesting ways to extend and automate STPA hazard analysis and requirement generation, supervised by Nancy. G. Leveson [26].

2.3.1 Context Variables

Step 1 of the STPA analysis want you to identify control actions that may violate safety constraints and cause hazards, but without extending the process this Thomas argues that this process has been ad-hoc, only guided by the definition of what could cause a hazardous control action. This ad-hoc manner to identify hazardous control actions make it difficult to have proper knowledge of the context of the control action. As seen in table 2.2, the user will have to think about in what context the type of hazardous control action may be hazardous, and then describe it in such a way that this context might be understood. This way of identifying hazardous control actions also has the possibility to be ambiguous about what constitutes *too early* or *too late* [26]. To support this problem Thomas suggest decomposing the context in to variable and values, as seen in table 2.3. Using this decomposing of the context may be a powerful tool to help the user identify hazardous control actions. But while this method provides more guidance than ad-hoc methods, this approach still relies on the user's ability to identify the relevant context variables and values.

A context table for the train door example can be seen in table 2.4. This table shows in what context the door open command can be a hazardous control action. This is done by showing the three PMVs train motion, emergency and train position in columns with their possible values. The analyst may then by reading a row determine if the control action is hazardous in that context if it is provided at all, provided too early or provided too late. The functionality of this

Table 2.4: A example context table for identifying hazardous control actions, from Thomas [26]

Control Action	Train Motion	Emergency	Train Position	Hazardous control action?		
				If provided any time in this context	If provided too early in this context	If provided too late in this context
Door open command provided	Train is moving	No emergency	(doesn't matter)	Yes	Yes	Yes
	Train is moving	Emergency exists	(doesn't matter)	Yes	Yes	Yes
	Train is stopped	Emergency exists	(doesn't matter)	No	No	Yes
	Train is stopped	No emergency	Not aligned with platform	Yes	Yes	Yes
	Train is stopped	No emergency	Aligned with platform	No	No	No

table may be extended by providing the hazards that might cause it to be hazardous, instead of a yes or no answer. This table also is smaller than all combination of variables and values, and uses “(doesn't matter)” as value, this is due to automation and simplification techniques discussed in section 2.3.3

2.3.2 Traceability for STPA

By using hazards to identify hazardous variables, and these variables to introduce values, you create a traceable process model hierarchy. This process model hierarchy could be used to make the method more traceable, allowing the analyst to get a better understanding of what other changes needs to be looked at when there is a change in the system.

2.3.3 Automating and Simplifying STPA

As the system being analysed increases in complexity, the process of analysing it also increases in complexity and effort. When the system grows more complex, not only does it increase the possible control actions that need to be analysed, it also increases the amount of process model variables and their values, that need to be evaluated for each control action. As found in the pre-study of this thesis, not using the right tools and techniques might make these powerful tools suggested by Thomas and Leveson not available for complex or agile systems [12]. Thomas suggests three techniques for improving the scalability of the methods suggested in his dissertation: abstraction and hierarchy, logical simplification, and continuous process model variables.

STPA is a top-down approach that makes use of abstraction and hierarchy, it is useful to use this to simplify the system. Under analysis a control action may be used on a lower level, but may be abstracted as to represent a composition of several control actions on a lower level of the system. This may also be used for the context variables and values, where a “emergency”

might be all that is needed for a higher level, but a lower level might need to separate the different types of emergency. Thomas also suggest that this abstraction will provide benefit to the early phases of design [26].

Logical simplification is the process of removing rows and values from the context table with “doesn’t matter” terms. Some values or combination values makes the other values, or the entire row invalid or always one state, making it possible to remove them from the table.

The last technique, continuous process model variables, discusses the possibility of reducing complexity by evaluating what values are important. A continuous value may have an infinity possible values, but the user often, after careful consideration, may reduce these to a small number of finite values. Reducing the amount of possible values, the variables may have, will drastically reduce the number of rows in a context table.

Thomas also suggests the possibility of using rules and automatic tools for automatically filling related hazards in the context table. These rules may be such that when a variable is one value, a control action will always cause hazard, such as opening door when train is moving in the train door example.

2.3.4 Agile Software Development Using STPA

As discussed in this chapter, STPA really on a hierarchical structure, making abstraction possible. This makes it more viable for using STPA analysis in agile software development environments, as it does not necessarily need up-front design [26], which is not preferred for agile development, such as Scrum [6]. In their article Wang and Wagner suggest mapping STPA into a Scrum method called Safe Scrum [28]. They suggest that one way of doing this is to use the control structure (architecture) as a cross point between STPA and Safe Scrum.

It was also found in the pre-study for this thesis that there would need to have proper traceability in the STPA analysis in such an environment, as one of the principles of the agile manifesto is that requirements change even late in development [2]. Without proper traceability and tools in a STPA analysis, such changes would make it hard for a user to understand how a change would affect the analysis.

2.3.5 System-Theoretic Process Analysis - Security(STPA-Sec)

System-Theoretic Process Analysis - Security is a suggested analysis method for security analysis based on STPA analysis [30] [29] [31]. STPA-Sec is an extension of STPA that thinks of security incidents as *intentional* disruptions, instead of the *unintentional* disruptions of safety accidents.

Young and Nancy argues that the difference between how a security expert and a safety expert see themselves is that safety experts try to prevent losses from *unintentional benevolent* actors, while security experts prevent losses due to *intentional* actions by *malevolent* actors [30]. This focus on intent may be misguided as you may never know the intentions of the *malevolent* actor, so the majority of energy and analysis should be refocused on loss prevention strategies.

STPA-Sec follow the same two step structure of STPA, where the analysis starts by defining the fundamental information redefined as security issues as seen in table 2.5.

Table 2.5: STPA to STPA-Sec

STPA	STPA-Sec
Loss	Accident
Hazard	Vulnerability
Unsafe Control Action	Unsafe/Unsecure Control Action
Safety Constraints	Security Constraints

2.4 Current STPA Tools

There already exists several software solutions for STPA that are in different stages of development [3]. They all implement the STPA analysis in different ways, but in the preliminary study they were found to maybe be lacking if they are to be used in an agile software development environment [12].

2.4.1 XSTAMPP using the A-STPA plugin

A-STPA is a tool that was developed as part of a student project to further the participants understanding for STPA, while trying to automate the process of the STPA approach [1]. This tool was then integrated as an included plugin in the XSTAMPP project [23]. As such in its

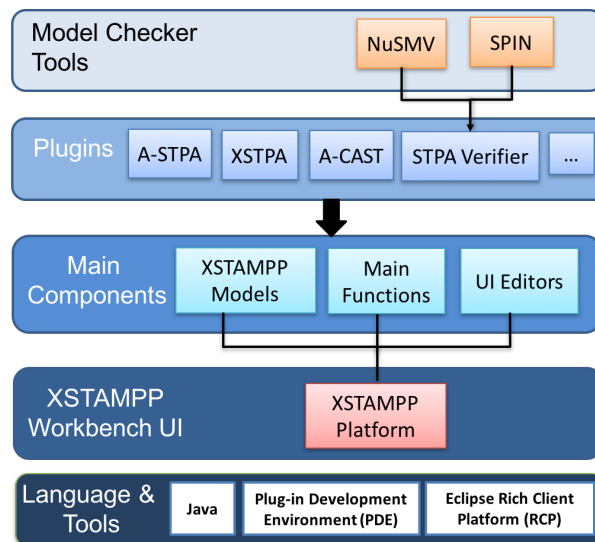


Figure 2.3: The XSTAMPP architecture from their homepage [23]

current most accessible form A-STPA is using the core functionality of XSTAMPP, while providing the necessary changes to implement the STPA approach. They are both based on the eclipse development tools Plug-in Development Environment (PDE) and Rich Client Platform (RCP). A figure showing the architecture of this is shown in Figure 2.3

XSTAMPP itself is a project to support STAMP, and supports the CAST, STPA, STPA-Sec and XSTPA approaches from STAMP, using different plugins.

XSTAMPP and its related plugins are free applications accessible from their website, and supports the Windows, Macintosh and Linux platforms [24]. For the windows platform this download provide a stand-alone product, that requires the Java Runtime Environment 7 or above. Macintosh were tested on one device, but failed to run without compiling their source code. As this was only tested on one Macintosh device, this may not be indicative of any problems with their provided support.

Results from previous work

Of the software solutions tested in the preliminary study for this thesis, using XSTAMPP was found to be the currently best solution. While it allows for some of the techniques introduced by Thomas, it mainly relies on the original STPA analysis. This makes analysis using this tool more ad-hoc [26], text heavy and harder to automate.

2.4.2 SafetyHAT Project

The SafetyHAT Project are an implementation of the STPA method, by using a custom Microsoft Access template. Microsoft Access is a database management system (DBMS), that provides support for graphic user interface(GUI). Using this SafetyHAT implement an approach for implementing STPA using graphical elements to provide step by step guide for the user to provide the necessary information, that then are stored as a database format. The software support export this database information to the Microsoft Excel format, by using database queries that are stored in different pages in the Excel document.

Results from previous work

While SafetyHAT is made for STPA, it does not provide much functionality to the analyst, that is not available using a spreadsheet software. It also uses standard STPA making it harder to use automation and simplifying techniques, and requires that the analyst describe the full context for each hazardous action.

2.4.3 Sahra

Sahra is a software that are in development, that promise to provide a hierarchical way to implement control structure, and provide full support of both steps of the STPA process [4]. As this software are still in development, and not publicly available as of the writing of this thesis, this tool is not taken into further consideration.

2.4.4 RM Studio

RM Studio is a risk management software [19], that have a module for STPA in developments [20]. They provide a demo video that indicate that this is a custom implementation STPA not strictly following the original STPA as suggested by Leveson, but providing the tools to follow it.

2.4.5 STAMP Workbench

Previously code name iStamp, Stamp Workbench is a Japanese developed software released March 2018 [15].

2.4.6 SpecTRM

SpecTRM is a tool provided by Safeware Corporation that supports STPA [3].

2.4.7 An STPA Tool

“An STPA Tool is a unpublished STPA tool created as partial fulfilment of the requirements for the degree of master of science”, by Daijiang Suo [21] [3] [27]. It is based on the eclipse integrated development environment, and implements automation as suggested by Thomas.

2.5 Incremental Software Development

To discuss research question two, we must first present what Agile development is. Agile software development describes a set of principles that aims to be an alternative to documentation driven software development processes. This is done through prioritising the values of [2]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

These values were decided on by a cooperation of representatives from several development processes that aimed to be alternatives to document driven development. As this is the case, agile software development is not itself a complete development approach, but rather represents a group of development approaches. Common approaches that is used by this group to facilitate these values, are incremental development, short feedback loops and direct cooperation with customer. The main focus of this report will be the goal of responding to change.

Chapter 3

Research Method

In this chapter will presented the research methods that was used in this thesis, the motivation for this method, and why the specific research questions were chosen.

3.1 Motivation

As STPA is made for more complex software based systems, there is the possibility of the analysis supporting the security and safety analysis of full software development. But current software assisting tools might be lacking the correct functionality to support an agile software development, as they may have been made for system using other system development methods than the ones used by agile software development. To look further into this, it was decided to implement a prototype that would implement functionality that was found to maybe have some positive affect on such a situation.

3.2 Choice of Research Questions

The research question was chosen to provide the most contributions from this project. Answering research question 1 would provide a list of requirements that may be used by any project trying to implement such a solution, and as such would not need the other research questions answered to provide a contribution.

Research question 2 was chosen to present the implementation used by this thesis to try the functionality that was suggested.

Answering research question 3 will provide a qualitative look into the current state of STPA

assistance tools, and what might be done to improve them.

The research question is the following:

RQ1 What requirements would be needed for an STPA application to support an agile development environment?

RQ2 How could the requirements presented in RQ1 be implemented?

RQ3 How would such a tool compare to current solutions for STPA analysis?

3.3 Method

Using the model of research process defined by Oates [14] this thesis will follow a design and creation strategy. Then by using information from other current solutions and experiences, there will be performed a qualitative comparison of the current solutions with the implementation that was created with the strategy.

As the main motivation behind this thesis was the possible functionality that a software tool could provide agile development using STPA analysis, there was a need for a qualitative analysis, as a quantitative analysis using groups might yield indecisive results when using a method that the users would have no prior knowledge of, on an unstable platform. For such a situation a paper prototype may have been used, but this method gives more results related to user interface design than functionality [18].

3.3.1 Interviews

While there were no formal interviews performed, there were generated requirements from informal discussions with the supervisor for this project and other students writing papers about STPA.

3.3.2 Software Prototype

As a strategy for this thesis there will be designed and created a software prototype that will be used to compare and understand the limitations of current solutions for performing tool assisted STPA analysis. The requirements for this tool will help answer research question 1, while the implementation will help answer research question 3.

3.4 Limitations

This thesis will focus on step 1 of the STPA analysis, as this step identifies what control action might be hazardous, and could be useful information that don't directly affect the development, but applying the requirements and constraints defined in step 2 of a STPA analysis might need more testing before applied in agile software development.

While security is important for software development, as presented in section 2.3.5, STPA-Sec is a young analysis that should be able to be performed with the same tools as standard STPA analysis. As such this will mostly infer that if something works for STPA, it would also work for STPA-Sec, and only mention cases where this is found to not be true.

Chapter 4

What functionality could a software system provide the STPA method?

In this chapter we will discuss how a software tool might support a safety expert performing a STPA analysis. To discuss this, we will use the literary review done in chapter 2 - State of the Art, the findings of the pre-study of this project [12], and look at what current solutions are doing. The control structure is separated into its own section as it could be made in different steps of the analysis and the tools needed for it is quite different than the other steps in establishing fundamentals. The chapter will first be divided into different steps that are needed for STPA, then we will look at how a software could support making changes after initial analysis. Lastly, we will look at the possibility of implementing specific support for the top-down hierarchical structure of STAMP, using it for adding abstraction, and providing a more readable structure. Summary tables for the suggested tools will be provided, with prioritized functionality, but they will be redefined as proper requirements in section 5.1.

4.1 Establishing Fundamentals

For defining the fundamentals of STPA you need to define the fundamentals, such as goals of the system, accidents, hazards, process model variables and control actions [8]. These actions may be seen as strict list management, limiting in what ways writing these may be supported. But lists in establishing the fundamentals are not independent lists that does not interact with each other or later steps. As such the support tools should provide tools for making the process of keeping these lists connected, such as linking accidents and hazards. If the extension of

STPA is used, it is also suggested to have a proper traceability from hazard to processes model variables to the hazardous control actions [26], where a software solution could provide the necessary views to make this traceability easier for the end user.

As mentioned establishing fundamentals is about list management, but there are some tools for managing list that could provide support for this, such as search, filtering and sorting, but if the analyst uses a hierarchical structure for the control structure, these lists should maintain a manageable size, limiting the usefulness of such tools. But as discussed in section 4.5 software tools could support establishing fundamentals for such structure.

For newer users of the STPA analysis method, there is also the possibility for a software tool to add guidance during the first steps of the analysis.

Table 4.1: Identified possible support functionality, for establishing fundamentals

Name	Description	Priority
List management	Search, filter and sorting	Medium
Traceability	Providing information and views that provide a proper trace through the process	Medium
Guide	End user might not be experienced with STPA, so guidance might prove useful	Low

4.2 STPA Step 1

With the fundamentals established, the next step of STPA analysis is to combine this information to identify the HCAs. To support the not extended method for STPA, the analyst would need a table to analyse the control actions to what could cause them to be hazardous. Supporting this method is current solutions such as XSTAMPP and SafetyHAT, and as seen in these applications does not need to provide much software support to the analyst other than preparing a table to compare, write context, and link HCAs to hazards. There are some arguments that linking each HCA to a high-level hazard is not necessary, as these actions are hazards themselves, but are suggested to help define these hazards [25] [7], but a software support tool should probably still support linking HCA and hazard. As HCA may be hazards themselves, there is also the possibility that functionality for redefining a HCA as a hazard or adding the HCA to the high-level hazards.

But if the end user uses the more formal methods presented by Thomas, there are more tools a STPA software could provide an analyst. First a support tool should support generating a context table as described by Thomas [26]. This table should include all permutations of process model variables values and control actions, so that an analyst could do HCA identification on all possible contexts. As this could generate a large table of possibilities, a support tool could also provide the possibilities to simplify the table through automation and/or manual tools. As discussed in section 2.3.3 Thomas suggest several ways that such simplification and automation could be done, where hierarchy and abstraction will be discussed in section 4.5.

Logical simplification is a technique that may have a big impact on the STPA analysis while maybe not be hard to implement at a base level. At a base level it could be implemented as buttons that removes rows from the context table, and buttons for setting PMV values as “doesn’t matter”. While this method would require the analyst to still go do a review of these values and rows, it might provide a fast way to remove a lot of possible HCA, without much work. Another possibility for such a functionality would be for adding automated removal of PMV values and rows based on rules provided by the analyst.

While Continuous Process Model Variables might be a useful technique for STPA simplification [26], it was not identified any tools a software could provide this technique, other than providing information that might support the analyst.

Providing the possibility for the analyst to provide rules for evaluating big tables could with proper rules let the system fill large portions of a context table, lessening the load of a STPA analyst.

Table 4.2: Identified possible support functionality, for STPA step 1

Name	Description	Priority
HCA table	A software tool could generate a fillable HCA table for standard STPA	Medium
Context table	A software tool could generate a fillable context table for extended STPA, as suggested by Thomas	High
Linking HCA to Hazards	Providing a proper view for linking HCA with hazards	Medium
HCA to Hazard	There could be a use for a functionality for adding or re-defining a HCA as a hazard	Low
Logical Simplification	Simplifying a context table by making HCA lines and PMV values to “doesn’t matter”	High
Continuous Process Model Variables	Support simplifying PMV values to less possibilities	Low
Hazard Rules	Filling the context table automatic by allowing end user to define control action and context rules	High

4.3 Control Structure

A control structure may provide or provided with most of the information needed for fundamentals, and generating context table [26]. As such allowing information to be transferred between a control structure and the information tables in an application could help support analysis, and ensure that an analyst not needs to provide the same information two times.

Most current solutions try to provide a proper solution for implementing a control structure [3], but as STPA is a relative young method, there might exists better design tools for control structure than one made specifically for STPA. STPA use generic designs and symbols, as such other design tools should support them, and if such a tool provided a format that might be interpreted by a STPA software, importing and exporting information to such a software might be a more suggested solution than implementing a design tool specific for STPA.

Table 4.3: Identified possible support functionality, for control structure

Name	Description	Priority
Fundamentals from Control Structure	A lot of the information needed for fundamentals and generating context table may be generated from a control structure	High
Importing or exporting Control Structure	Letting a user import and export control structure information such as fundamentals from other applications	-

4.4 Agile software development team

If STPA analysis is done in the start of a project, there will probably need to made changes to the analysis as the project evolves and changes, especially if the project is and agile development environment that has the principle that change will come even late in development [2]. But as STPA is linked method, such change may have affected other parts of the analysis or system, that the analyst does not know of. By understanding STPA, providing proper traceability and linking in the process a software tool might provide information and suggestions for what other part the analysis needs to be re-evaluated when such change is made. This suggestion might be in the form of a list of elements to look at, or colouring elements in the system that should be looked at.

If the analysis is managed by multiple people, even with change suggestion it might not be easy for everyone to keep track of the changes. One solution for this could be for the system to have some form of version control to help with functionality as version history, changes and branching. For a team project to work there could also be a use for team management support, such as roles and letting multiple people access the same project at the same time.

Table 4.4: Identified possible support functionality, for change management

Name	Description	Priority
Re-evaluation suggestion	Providing the analyst suggestions for what needs to be re-evaluated when making changes	High
Version control	Supporting team analysis by providing the tool provided by a version control system	High
Team management	Functionality such as roles and access for multiple users	High

4.5 Hierarchical Structure

While STAMP is a hierarchy structure, STPA analysis does not necessarily need to be hierarchical, but as suggested by Thomas, following a hierarchical structure may allow to simplify the analysis with abstractions [26]. A software for STPA analysis could have use of looking at this type of STPA analysis, and provide tools for supporting it. For STAMP hierarchy it is very important for there to be proper communication between the levels of the structure, so that constrains may be properly be enforced. A software tool could use this information to ensure that information is properly abstracted and connected between the different levels, and warn the user for levels that are missing links.

Table 4.5: Identified possible support functionality, for hierarchical STPA analysis

Name	Description	Priority
Communication	To support hierarchical STPA analysis there should be some tool for providing communication between the different levels, and let the different levels affect each other.	Medium
Abstraction views	There should be tools for the analyst to not lose control over the different levels	Medium

Chapter 5

Results of Research Question 1

This chapter presents the architectural requirements that were derived from the findings in chapter 2 - State of the Art and the preliminary study. First will be presented the goals that the system will aim to achieve. Then using these and previous findings will be derived the functional and quality requirements of the system.

5.1 Architectural Requirements

For this system it was used goal driven requirement development method to gather the requirements. First goals for the system will be defined, and then using a top down approach the sub goals, and requirements will be defined. And using these requirements to identify changes to goals in a bottom-up approach.

5.1.1 The Goals of the System

- Supporting STPA analysis
 - Supporting STPA step 1 analysis
 - Providing a tool using automation techniques as suggested by Thomas [26]
- Providing tools for agile software development
 - Version control
 - Team project

- Provide tools found in chapter 4 - What functionality could a software system provide the STPA method?
- Test what a STPA software tool could do

5.1.2 Functional Requirements

The functional requirements (FR) of a system are the requirements that defines what functionality a system should be able to provide. These requirements where derived from the goals of the system, and the earlier findings of this thesis and the preliminary study.

Storage

The system should provide a non-runtime storage for information provided by the user or generated by the system.

Table 5.1: Database Requirements

req. no	Name	Description
FR 1.1	Database	The system should have a database
FR 1.2	Manage data	At runtime the system should be able to modify, read and write data to the database.

User Profile

For the system to receive and provide information specific to the user, the system would require some form of profile system as a cross point between system and user.

Table 5.2: User Profile Requirements

req. no	Name	Description
FR 2.1	User	A user of the system should be able to create or delete a user profile.
FR 2.2	Project	A user should be able to create or delete a project they control.

Authentication

For the system to receive and provide information specific to the user, the system needs to be able to authenticate the user.

Table 5.3: Authentication Requirements

req. no	Name	Description
FR 3.1	Authenticate the end user	The system should provide some form of authentication method for identifying the user.
FR 3.2	Login and logout	A user should be able to start and stop being authenticated.
FR 3.3	Login required	For sections of the system that requires authentication, the system should provide a login required message.

STPA fundamentals

To achieve any of the goals suggested the system would require providing the user the ability to define the fundamental information needed for a STPA analysis. The system should also add functionality for supporting this more than would be done by pen and paper.

Table 5.4: STPA Step 1 Requirements

req. no	Name	Description
FR 4.1	Fundamentals	The User should be able to provide the system with the fundamental information for STPA step 1.
FR 4.2	Modify	The User should be able to modify all information provided.
FR 4.3	Linking hazards and accidents	The user should be able to link accidents and hazards
FR 4.4	Search, filter, tagging and sort	The system should provide list management functionality such as search filtering, tagging and sorting
FR 4.5	Guide	The system should be able to provide guidance information for inexperienced STPA analysts

Extending STPA

To make the software better suited for the goals of the system, it should provide the functionality presented in section 2.3. To support the extension of STPA that is suggested, the system would need to support a context table, as shown in table 5.5, and automation as presented in

table 5.6. The requirements for these is identified as the same subset of requirements, while agile support as part of goal driven requirement has been moved to its own subset of requirements.

Table 5.5: Context Table Requirements

req. no	Name	Description
FR 5.1.1	Process Model Variables	The user should be able to define process model variables, and their possible values.
FR 5.1.2	Context table	The system should be able to present a context table as defined by Thomas [26].
FR 5.1.3	Hazards	The user should be able to link hazards to hazardous control actions.

Table 5.6: Automating and Simplifying Requirements

req. no	Name	Description
FR 5.2.1	Abstraction and Hierarchy	The user should be able to create abstracted systems
FR 5.2.2	Logical Simplification	The system should provide support for logical simplification for the context table. The user should be able to remove rows, and set values as “doesn’t matter”
FR 5.2.3	Define hazard rules	The analyst should be able to define rules for HCA that automatic fills the relevant part of the context table

Agile Software Development Support

As suggested by the research question of this thesis and the goals of the system, the system should provide functionality for using STPA in an agile software development environment. These requirements were based on the findings in the preliminary study and in chapter 2 - State of the Art. While section 2.3.4 presents one method to use STPA in an agile environment, this system will not be limited by a specific method, and rather provide functionality that may be useful for an agile software development environment. For the requirements of this system agile software development will be thought of as a team project, and as such team functionality will be under agile software development.

Table 5.7: Agile Software Development Requirements

req. no	Name	Description
FR 6.1	VCS	The system should have version control to provide better support for teams.
FR 6.2	Feedback to changes	The system should provide feedback to what part of the analysis that needs to be re-examined.
FR 6.3	Project	A user should be able to be part of multiple projects, and be able to change what project is active.

Traceability

Table 5.8 presents the requirements for the system to provide traceability to the user, when the user use the system for STPA. Traceability requirements for this system will be seen as the functionality to trace history and items in the system related to each other for STPA analysis, and not the traceability for the implementation of the system. This system will see the control structure as a cross point between STPA and agile software development, ad suggested in section 2.3.4 see the control structure as a cross point between STPA and agile software development. While the system will not have functionality for designing control structure models, it will have requirements for interacting with such models.

Table 5.8: Traceability Requirements

req. no	Name	Description
FR 7.1	Control structure to analysis traceability	The system should provide the user with information between control structure and analysis in the STPA process.
FR 7.2	Design to analysis changes	There should be some solution to importing changes to the analysis process from control structure.
FR 7.3	Analysis to design changes	There should be some solution to importing changes in to the control structure from analysis.
FR 7.4	Traceability for hazard, PMV and HCA	The system should have a traceable process from hazard to HCA.
FR 7.5	Hierarchical	The system should provide communication between different levels of a hierarchical STPA analysis

STPA-Sec

In section 2.3.5 there were presented an extension of STPA that takes security into consideration. While security could be an important aspect of a software development cycle, STPA-Sec will not be much of a consideration for the requirements as the suggested method for STPA-Sec is about the mentality of the analyst and renaming the steps in STPA. As such there was not found many ways that a tool could specifically support STPA-Sec other than renaming the values in the tool for STPA.

Table 5.9: STPA-Sec Requirements

req. no	Name	Description
FR 8.1	Rename	The system should provide a way for the user to activate a STPA-Sec mode, where items are renamed as presented in table 2.5.

5.1.3 Quality Requirements

Quality requirements, often referred to as non-functional requirements, are requirements used to evaluate the performance of a system. Unlike functional requirements they do not specify specific behaviour or functionality of a system, but often specify an overall property of the system. This section will describe some of these requirements for the system.

Usability

A usability requirement describes how easy it should be for the user to accomplish a desired task or what kind of user support the system should provide [5]. This quality attribute was chosen as the main functionality for the system is supporting the user in using the STPA analysis. Such a tool is not necessary for performing such an analysis, so the value of the system comes from its ability to supporting this process.

U1: Supporting STPA Step 1 Analysis

These requirements compare the systems capability to support a STPA Step 1 analysis to other solutions. If users are to use the system, it should compare favourably to other solutions.

Table 5.10: Quality Requirement U1

Source of stimulus	End user
Stimulus	User starting new STPA analysis
Artefacts	System
Environment	Runtime
Response	The user has finished STPA step 1 analysis
Response measure	Time comparison to other solutions

U1: Supporting STPA Step 2 Analysis

These requirements compare the systems capability to support a STPA Step 2 analysis to other solutions. If users are to use the system, it should compare favourably to other solutions.

Table 5.11: Quality Requirement U2

Source of stimulus	End user
Stimulus	User starting STPA step 2 analysis on a pre-prepared step 1 analysis
Artefacts	System
Environment	Runtime
Response	The user has finished STPA step 2 analysis
Response measure	Time comparison to other solutions

U2: Easy to learn system features

New users would like the system more if it was easy to understand and start using it to its full capabilities.

Table 5.12: Quality Requirement U3

Source of stimulus	End user
Stimulus	End user visits website
Artefacts	System
Environment	Runtime
Response	The user has logged in and learned all the main features of the system
Response measure	Within 15 minutes of experimentation

Modifiability

The modifiability attribute is about how costly and risky it is to make changes. Modifiability is normally centred around the developers making changes to the system, but an end user changing a hazard in the system is clearly also making changes [5]. For this attribute both the developer making changes to the code, and an end user making changes to steps in the STPA analysis.

M1: End User Making Changes

To support an agile software development environment, it would be necessary for the system

to support changes to the STPA analysis at any time. To support this the system should make suggest other parts of the analysis the needs to be re-evaluated when making changes.

Table 5.13: Quality Requirement M1

Source of stimulus	End user
Stimulus	User makes change the STPA analysis
Artefacts	System
Environment	Runtime
Response	The user has made all follow up changes
Response measure	The time and accuracy for the user making follow up changes

M2: Developer Making Change to the System

As STPA is a newer analysis methodology, and still has ad-hoc methods [26], the system should be modifiable as to be possible to support possible extensions of the method such as the ones presented in section 2.3. One such extension is STPA-Sec.

Table 5.14: Quality Requirement M2

Source of stimulus	Developer
Stimulus	Change request
Artefacts	Module
Environment	Design time
Response	Made and tested the change
Response measure	Cost of change

Interoperability

Interoperability is about the degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context [5]. As this thesis will not suggest or implement a way for the user to design models for control structure, this quality attribute is chosen for measuring the system's ability to interact with control designs from other systems. The functional requirements for this functionality is presented in section 5.1.2.

I1: End User Initiates an Import or Export

This requirement is a measure of the systems capability to respond to a user requested inter-

action with a design from another system.

Table 5.15: Quality Requirement I1

Source of stimulus	End user
Stimulus	An import or export request
Artefacts	System
Environment	Runtime
Response	The import or export is processed
Response measure	Accuracy of the import or export

I2: System Support

This requirement is a measure of the systems capability to support interoperability with different systems.

Table 5.16: Quality Requirement I2

Source of stimulus	End user
Stimulus	Import or export request
Artefacts	System
Environment	Runtime
Response	The import or export is processed
Response measure	Number of supported systems, and time for end user to make the systems interact

Security

The security attribute is a measure of the system's ability to protect data and information unauthorized access [5]. While this is an important quality attribute, there will be no specific requirements for this, as the system implemented system presented in chapter 6 - Results of Research Question 2 is a prototype not intended for public use, and a full system implementation not necessarily will be in the same environment.

Chapter 6

Results of Research Question 2

This chapter will discuss the implementation of the requirements presented in chapter 5 - Results of Research Question 1. This chapter will be divided in to two sections, and will first in section 6.1 discuss the back end of the system, and in section 6.2 will discuss the front end. This separation between front end and back end is based on what layers of the system is presented to the user. As the implementation for this system is a web application the infrastructure is client-server based, the front-end is the processing done by the client, while the back-end is the processing done on the server. As the implementation will have no direct control over what software is used by the end user, the processing of information sent to the client will be seen as the front-end in this chapter.

6.1 Back End

The back end of a system is the layer that is concerned with managing and providing data. This data is not directed the end-user but used by the front-end to provide the end-user a view.

6.1.1 Framework

Flask is a micro web framework written in Python. Flask defines the “Micro” as that they aim to keep the core simple but extensible, as such the user could implement the web application in a single Python file, but does not necessarily need to [16]. This is done so that flask does not make many decisions for the user, and leave it to the user to add extensions for functionality such as database integration, form validation, upload handling and authentication as if it was

implemented in Flask itself [16]. As such the implementation of the system relies on several extensions for the framework, as is seen later in the chapter.

The system uses this framework to provide different pages to the user through blueprints, that provide blueprints for extending the application. These blueprints provide a modularity to the application while also make larger application more manageable by allowing the application to have a better file structure. An example of a blueprint is shown in listing 6.1 and listing 6.2, where it is shown a hazards blueprint being registered in `__init__.py` and defined in `hazards.py`.

```
1 hazards_blueprint = Blueprint('hazards', __name__,
2                               template_folder='templates', url_prefix='/hazards')
3
4
5 @hazards_blueprint.route('/')
6 @login_required
7 def index():
8     ...
```

Listing 6.1: Defining the hazards blueprint and index page

```
1 app.register_blueprint(hazards_blueprint)
```

Listing 6.2: Registering the hazards blueprint

6.1.2 Database

For database it was chosen to use the Object-relational mapper (ORM) SQLAlchemy for managing the SQL databases. This was done to make it easier to manage a SQL database from within the Python code, and manage the information as object-oriented (OO) objects. For choice of engine it was chosen to use SQLite as it was known by developer, it is Python built-in as the `sqlite3` module and the SQLAlchemy implementation as a layer between code and database, ensured that changing it at later date would be negligible, if needed.

The models were decided to be split into two separate databases, where one is duplicated as several database files based on the same schema as discussed in section 6.3.5. These two databases are the User database and the Project database, where there is a many to many relationship between them, that is not directly implemented in schema, but indirectly as an

abstraction table with project id in the User database and a folder for the Project database as seen in listing 6.3 and listing 6.4.

Figure 6.1 shows a simplified ER diagram for the two databases. In the user database there is the User table to keep track of possible users, the unused Role table, and the abstraction of the Project database the Project table.

In the Project database you have the Goal table, that is isolated as it is not directly used in any part of the STPA analysis, but was included as a table and a page as it was useful for testing, and could provide some indirect use for the STPA analysis. The other tables except HCA store information related to the STPA analysis, while the HCA table is a collection of this information for the context table.

```

1  class Project(Base):
2      __tablename__ = 'project'
3      id = Column(Integer(), primary_key=True)
4      title = Column(String(80))
5      description = Column(String(255))
6      users = relationship('User', secondary='project_users',
7                          back_populates='projects', lazy='dynamic')
8
9      def __init__(self, title, desc, user):
10         self.title = title
11         self.description = desc
12         self.users.append(user)

```

Listing 6.3: The Project table in the user database schema

```

1  class ProjectDB:
2      PBase = declarative_base()
3      def __init__(self, project_id):
4          path = 'sqlite:///resources/db/projects/{}/project.db'.format(project_id)
5          self.project_id = project_id
6          self.engine = create_engine(path, convert_unicode=True)
7          self.project_db_session = scoped_session(sessionmaker(autocommit=False,
8                                                                autoflush=False,
9                                                                bind=self.engine))

```

Listing 6.4: The start of the ProjectDB class, that takes a project id and provides a ORM for that project database

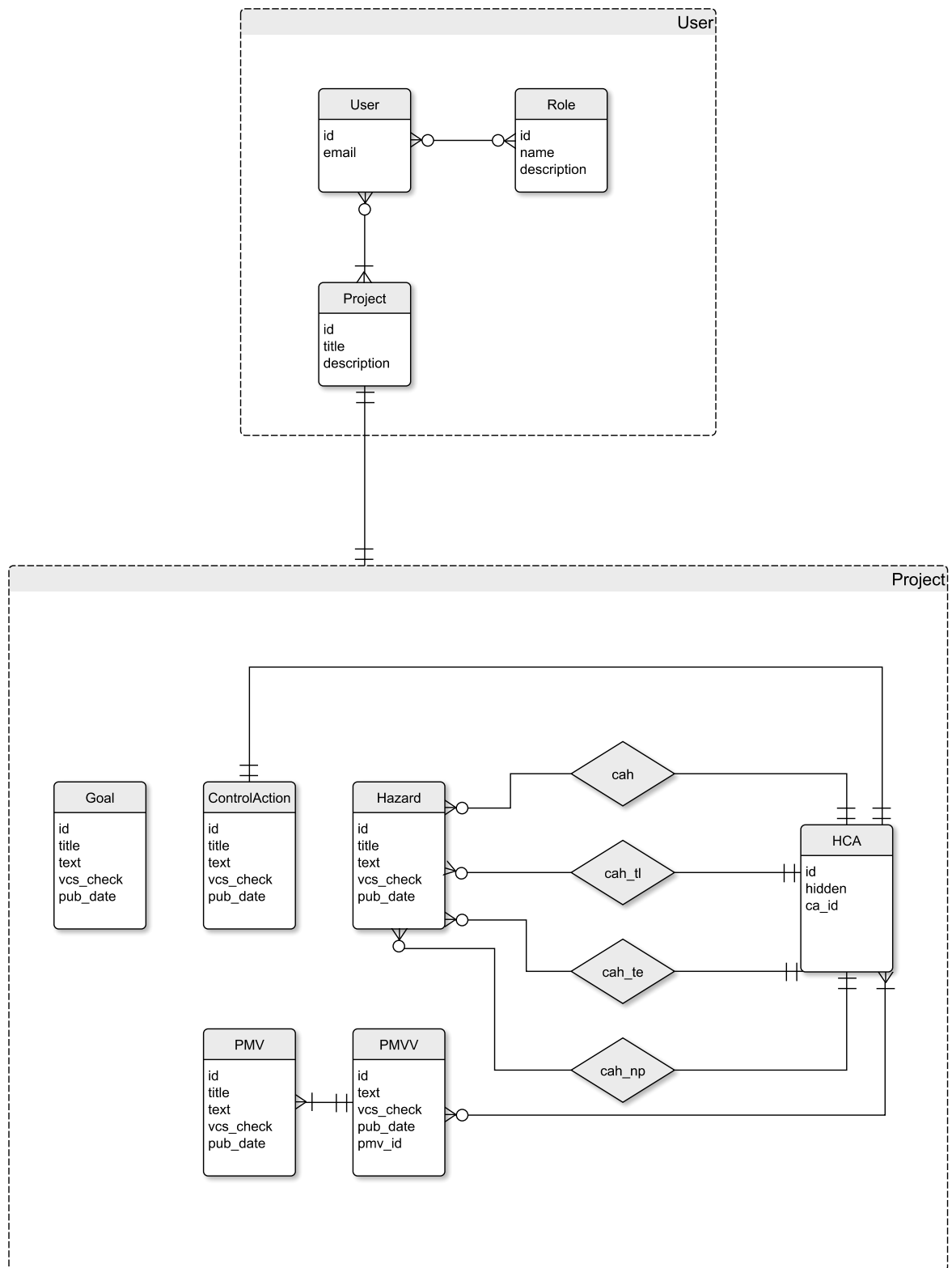


Figure 6.1: ER diagram of the databases

6.2 Front End

The front-end of the system is the presentation layer meant for the end user to see. In a web application, as implemented, the only control the system has for what is presented to the end-user, is what is sent by HTTP to the client. The processing of this on the client side may be done in any way, with any software that the end-user may chose. As such generating this information sent to the client, is the systems control over the presentation of the system.

6.2.1 Templates

As flask is based on the Jinja 2 template engine [17], it was used as templates in this project. Jinja2 is a template engine for python, providing tools for escaping standard html static code to provide Python like expressions. This provides the possibility for html to provide dynamic views, with data from running code.

While Jinja 2 is providing the Python like expressions WTFORMS is used for providing the Form input for the system. WTFORMS provide generation of HTML form fields, the validation of these fields, and the possibility to customize them in the templates. This allows the separation of code and presentation [22].

6.3 Requirements Implementation

In this section we will look at what was done to specifically fulfil the requirement specifications provided in chapter 5 - Results of Research Question 1, that were not fulfilled by the implementations described earlier in the chapter.

6.3.1 Storage and User Profile

The implementation of storage is presented in section 6.1.2, while the implementation of user profile is presented in section 6.1.2 and section 6.3.2

6.3.2 Authentication

Since the project has different users, that will access the same application, there were a need to implement an authentication system to manage the users. For this it was decided to use

The screenshot shows a web application interface for updating a PMV. At the top, there is a navigation bar with the following items: STPA, Projects, vcs, System Goals, Hazards, PMV, Control Action, HCA, Login, Register, and Logout. The main heading is "Show and Update PMV". Below this, there is a text input field containing "Train Motion". Underneath, there is a section titled "Add PMV Values" which contains two text input fields: "Train is moving" and "Train is stopped". A green button labeled "Add PMV Value" is positioned below these fields. Below the "Add PMV Values" section is a large text area labeled "Describe the pmv". At the bottom of the form, there are two buttons: a green "Create PMV" button and a blue "Back to list" button. The footer of the page reads "Prototype STPA application".

Figure 6.2: A view of updating a PMV and its associated PMV values

the Flask extension Flask-Security. As managing a Login functionality was the priority for an authentication functionality, the Flask extension Flask-Login was evaluated as a possibility. But as Flask-Security integrate this extension, and included other tools, it was evaluated as the extension to use.

Flask-Security provide the tools to generate a user for a SQLAlchemy managed database. Flask-Security also provides the tools for hashing and salting the passwords of the user. While this implementation works as a way to identify user in a prototype, there would need to be more prioritization for security in a system hosted on a public server.

6.3.3 STPA fundamentals

To let the analyst, establish fundamentals the front-end systems and back-end systems described in earlier chapters where used to let the user create list of hazards, PMV and PMV values.

A figure of updating a PMV and PMV values, may be seen in Figure 6.2

6.3.4 Extending STPA

By allowing the analyst to provide PMVs, PMV values and hazards the system generates a context table of all possible permutations of PMVs and PMV values. At the end of the table it is

#	CA	Train Motion	Emergency	Train Position	Hazardous	Too Late	Too Early	Not Provided	Hide HCA
1	Door open command	Train is moving	No emergency	Not aligned with platform	[1] [2] +	+	[1] [2] +	+	X
3	Door open command	Emergency exists	Train is moving	Not aligned with platform	[1] [2] +	[1] [2] +	[1] [2] +	+	X
5	Door open command	Train is stopped	No emergency	Not aligned with platform	[1] +	[1] +	[1] +	+	X
6	Door open command	Aligned with platform	Train is stopped	No emergency	+	+	+	+	X
8	Door open command	Aligned with platform	Emergency exists	Train is stopped	+	[5] +	+	[5] +	X

Generate table Show hidden hca

Prototype STPA application

Figure 6.3: A view of the context table for the train example

added four columns with hazardous if provided too late, too early, not provided and provided. When identifying a HCA, the analyst may then add hazards that the HCA may cause.

A figure of the implemented context table may be seen in Figure 6.3. A bug was discovered too late, where the PMV values does not align with the correct column. This does not change the analysis, as the correct values is in the correct row.

6.3.5 Version Control

To support the possibility of STPA analysis in a software environment it was decided to make the possibility of version control system(VCS) for the application. For this purpose, the VCS Git provided the necessary tools for implementing this.

As Git does not provide any direct support for python, GitPython was used for providing managing Git. GitPython provides an abstraction for git repositories, that enables the user to interact with them as if they were python objects, as seen in listing 6.5.

For the STPA Prototype each project is a different database file, and its own repository. This makes it possible for using the full extent of the Git toll set for each project, such as commit, merge, reset and branches. While not implemented this also opens the possibility for further authentication protocols making users have limited access to one branch, but full access to other branches.

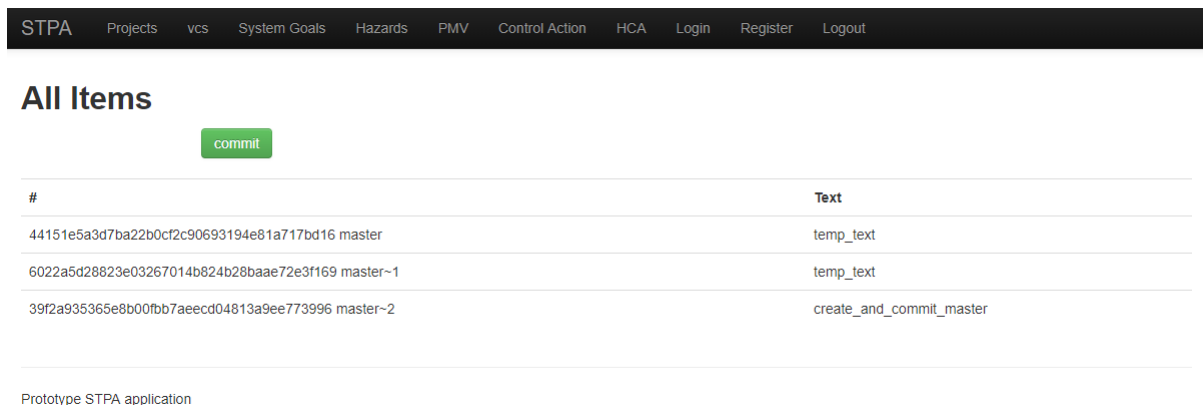
A figure of the implemented version control may be seen in Figure 6.4.


```

1 def create_and_commit_master(project_id):
2     repo_path = join(os_path, str(project_id))
3     repo = Repo(repo_path)
4     assert not repo.bare
5     index = repo.index
6     index.add(['project.db'])
7     index.commit('create_and_commit_master')
8     repo.create_head('master')

```

Listing 6.5: Using GitPython create a new project, and adding an initial commit



#	Text
44151e5a3d7ba22b0cf2c90693194e81a717bd16 master	temp_text
6022a5d28823e03267014b824b28baae72e3f169 master~1	temp_text
39f2a935365e8b00fbb7aeecd04813a9ee773996 master~2	create_and_commit_master

Prototype STPA application

Figure 6.4: A view of the current Git commits

6.3.6 Traceability

As the system uses the extended version of STPA, it provides traceability from process model variable to HCA. The system also allows the user to display the HCAs that are related to a hazard.

6.4 Implementation Result

This section will present an overview over the status of the implementation in relation to the requirements provided in chapter 5.1 - Architectural Requirements. Quality requirements will not be covered as they were not properly tested for this implementation, but they should be tested for a full implementation of the system.

Table 6.1: Functional requirements covered by the implementation

Req. nr	Implemented	Comment
FR 1	Yes	-
FR 2	Yes	-
FR 3	Yes	-
FR 4	Partial	FR 4.3-4.5 was not implemented due to time restrictions and prioritization
FR 5.1	Yes	-
FR 5.2	No	Partial implementation of FR 5.2.2
FR 6	Partial	FR 6.2 was not implemented due to time restrictions.
FR 7	No	Partial implementation of FR 7.4
FR 8	No	Not implemented due to time restrictions and prioritization

6.4.1 FR 4.3-4.5

FR 4.3-4.5 was not implemented due to time restrictions and perceived low priority. As these are functionality that does not directly remove work from the analyst, interviews and questionnaires should be provided for properly prioritizing and implementing these. FR 4.3 is an important step of STPA for unexpanded STPA, but down-prioritized in the extension suggested by Thomas, and was not prioritized for implementation, and cut due to time restrictions.

6.4.2 FR 5.2

FR 5.2.1 and FR 5.2.3 was not implemented due to time restrictions, but as they are identified as medium and high priority they should be implemented in a full system. FR 5.2.2 was implemented in a basic state where the analyst may use buttons in the context table to remove rows and values.

6.4.3 FR 6.2

FR 6.2 was prioritized highly and ended up not being implemented due to time restrictions, but should be implemented for a full system.

6.4.4 FR 7

Cut due to time restrictions, but a partial implementation of FR 7.4, where the analyst is provided traceable information through the hazard, PMV and HCA steps.

Chapter 7

Comparison of current software tools

In this chapter we will take a look at the current tools for assisting STPA analysis, and contrast them to better understand what could be improved, and understood from the different solutions. We will use the findings in chapter 4 - What functionality could a software system provide the STPA method? and the functionality of the current solution as a baseline for what is possible for a tool to assist the analyst with. We will also take an estimate look at cost for the different solution based on actions needed, and automation provided. The solutions that will be looked at is the ones discussed in section 2.4. As several of the solutions are in development or not accessible, it will be discussed based on perceived functionality and tools. The SpecTRM and Sahra solutions was not accessible to test for this comparison.

7.1 Compared versions

This section will present the version of the solutions that was tested for this comparison.

7.1.1 STPA Prototype

The STPA prototype was implemented as part of this thesis, and was compared with the last version of the project as of the writing of this thesis 26 June 2018. This version is a prototype application that is in no state of full release, and is developed to provide a testing platform for suggestions provided in this thesis.

7.1.2 XSTAMPP

XSTAMPP was tested with version 2.5.0, as the website was down as of June 2018, and this was the version that had been downloaded by the author. Where there is different version accessible by sourceforge and two github accounts.

7.1.3 Stamp Workbench

Stamp Workbench was tested with version 1.0.0/5c1123, which was the newest public accessible version as of 21 June 2018.

7.1.4 SafetyHAT

An unknown version of SafetyHAT was tested for this chapter. The setup file indicates a last modification as of 27 March 2014.

7.1.5 An STPA Tool

There was no published version of this tool, as such all comparisons for this tool is based on what is written in the thesis paper this tool was implemented for [21].

7.1.6 SAHRA

SAHRA is still in development, and provide no public accessible version, and will be compared based on information they provide on their web page [4].

7.2 Establishing Fundamentals Suggestions

As previously discussed, establishing the fundamentals is mostly providing the analyst lists, and is necessary for STPA analysis, as such all solutions support this as a base functionality. Table 7.1 provide an overview of how the current solutions support the suggestions, while the rest of the section will describe how it they are supported.

Table 7.1: Establishing fundamentals tools overview

Suggestion	STPA Proto- type	XSTAMPP	Stamp Workbench	SafetyHAT	An STPA Tool
S 1	Full	Full	Full	Full	Full
S 1.1	No	Partial	No	Partial	Unknown
S 1.2	Partial	Partial	Forced	Partial	Unknown
S 1.3	No	Partial	Partial	Partial	Unknown

7.2.1 List management

Only XSTAMPP and SafetyHAT providing some support for managing lists. XSTAMPP provide the analyst the possibility of moving items up and down in the list, and have a basic filter that checks the item for containing the provided word, hiding other items. SafetyHAT provide some list management by allowing the analyst to order items by the time they were entered or by name in alphabetical order.

7.2.2 Traceability

Most of the current solutions support some basic traceability through the STPA analysis.

STPA Prototype as presented in section 6.3.6 provides some implemented traceability, but not enough to be a full implementation of the suggestion.

XSTAMPP provide the user the ability to link hazards, accidents and constraints. It also provide synchronization between control structure and the HCA table by allowing the analyst to define control actions in the control structure. It also allows filtering the HCA table based on different fundamentals allowing the analyst to get an table that show only HCA related to a hazard [25].

Stamp Workbench provide a direct forced link from accident to hazard and safety constraints, not allowing the analyst to define hazard or safety constraint if it is not connected to a accident. The application also allow user to define control actions that are in control structure and HCA table, but does not provide a way for the user to trace hazard to HCA table, and only indirectly connects hazard to HCA by safety constraints.

7.2.3 Guide

While none of the applications support a full teaching tool, XSTAMPP, Stamp Workbench and SafetyHAT provides help information that tells the user how to do the STPA process and how to use the application.

7.3 STPA Step 1

As the first step is an important step of the STPA analysis, all current solutions support this step fully, but by different techniques. The biggest difference in techniques is if the application use the Systematic Method as suggested by Thomas, or follow the original method as presented by Leveson. An overview of the support for this suggestion may be seen in table 7.2.

7.3.1 HCA Table

XSTAMP, Stamp Workbench and SafetyHAT use the original method as presented by Leveson, as such support this functionality. But SafetyHAT does not provide a proper table and instead let the analyst look at individual HCA as list items.

7.3.2 Context Table

STPA Prototype and An STPA Tool is the two applications using this functionality, where they generate full context tables for the analyst based on information provided earlier in the analysis.

7.3.3 Linking HCA to Hazards

All solutions presented in this chapter supports functionality for letting the analyst get a view of possible hazards to add, and adding them to the relevant HCA type. While all support it Stamp Workbench provide support for this indirectly by linking safety constraint instead of linking hazard.

Table 7.2: STPA step 1 support

Suggestion	STPA Proto- type	XSTAMPP	Stamp Workbench	SafetyHAT	An STPA Tool
S 2	Full	Full	Full	Full	Full
S 2.1	No	Full	Full	Partial	No
S 2.2	Full	Partial	No	No	Full
S 2.3	Full	Full	Partial	Full	Full
S 2.4	No	No	No	No	No
S 2.5	Partial	Partial	No	No	Full
S 2.6	No	No	No	No	No
S 2.7	No	No	No	No	Full

7.3.4 HCA to Hazard

None of the solutions discussed in this chapter provide any support for helping the analyst redefining a HCA as a Hazard.

7.3.5 Logical Simplification

As logical simplification is a technique specific to the extended STPA method, STPA Prototype is the only one that partially supports this method. It supports it by providing the analyst the ability to hide HCA rows that is not necessary for the analysis.

7.3.6 Continuous Process Model Variables

None of the current solutions provides any specific assistance to the analyst, for limiting the number of values a process model variable may have.

7.3.7 Hazard Rules

While not tested, An STPA Tool is supposed to fully support making rules for automatic connecting HCA to hazards in the context table.

7.4 Control Structure

Most of the solutions presented in this chapter supports the creating of a control structure, while SafetyHAT and STPA Prototype relies on the analyst providing a control structure from another application. SafetyHAT allows the user to link a control structure model, but do not use

Table 7.3: Control structure support

Suggestion	STPA Proto- type	XSTAMPP	Stamp Workbench	SafetyHAT	An STPA Tool
S 3	No	Full	Full	Partial	Full
S 3.1	-	Partial	Partial	-	Unknown
S 3.2	No	-	-	No	-

this other than to allow the analyst to quickly access a view of the control structure, but as part of the analysis SafetyHAT asks the analyst to define a control structure by linking components as list items. A overview of the support for this suggestion may be seen in table 7.3.

7.4.1 Fundamentals from Control Structure

Depending if the solution implements the systematic method or the original method for STPA analysis, there are differences for what information a control structure could provide the system.

XSTAMPP are able to use information about control actions between the control structure and the HCA table, allowing the analyst to only define control actions once. As the solution also have a partial implementation of the systematic method for causal analysis, it also allows the analyst to define process model variables in the control structure.

Stamp Workbench allows the user to define control actions in the control structure, that is later used by the HCA table.

An STPA Tool while not tested, as the solution uses the systematic method, allows a analyst to define control actions and process model in control structure, one would assume this information is used by the solution.

7.4.2 Importing or Exporting Control Structure

As this is a functionality suggestion for solutions that does not implement its own control structure, it is only relevant for STPA Prototype and SafetyHAT, but none of the solutions support it. SafetyHAT allows the user to link a control structure, but does not use information from it. For STPA prototype this was a planned feature, but was cut due to time restrictions.

Table 7.4: Support for agile software development teams

Suggestion	STPA Proto- type	XSTAMPP	Stamp Workbench	SafetyHAT	An STPA Tool
S 4.1	No	No	No	No	Unknown
S 4.2	Partial	No	No	No	Unknown
S 4.3	Partial	Partial	No	No	Unknown

7.5 Agile software development team

For the solutions presented in this chapter, there are currently not much agile software and team management support, but as all applications except STPA Prototype are local applications, a team could manage the files them self. A overview of the support for this suggestion may be seen in table 7.4.

7.5.1 Re-evaluation suggestion

This suggestion might be useful for any analysis that expect changes, such as in an agile software development environment, but none of the current solutions has such functionality.

7.5.2 Version control

STPA Prototype is the only solution that currently directly support any form of version control, but this support is in a stage where it mimics the functionality of saving a file, but as it uses git as it version control tool, it should be able to support more version control functionality with some work.

As XSTAMPP store its save files in xml format, a team could manually add the file to version control software and get most of the potential from version control with it.

7.5.3 Team management

XSTAMPP supports team management by allowing the assignment of roles for a project, where a user may be an admin or user, and a user may be restricted to read only access.

As STPA Prototype is a web application it should support multiple users accessing a project and making modification at the same time, but this functionality is not properly tested. The database support roles, but functionality for such feature have yet to be added to the application.

Table 7.5: Support hierarchical structure

Suggestion	STPA Proto-type	XSTAMPP	Stamp Workbench	SafetyHAT	An STPA Tool
S 4	No	Partial	Partial	Partial	Partial / unknown
S 4.2	No	Partial	Partial	Partial	Partial / Unknown
S 4.3	No	No	No	No	No

7.6 Hierarchical Structure

The in development solution sahra is the only solution that try to directly support the possible hierarchical structure of STPA [4]. As all solutions except STPA Prototype support control structure, that allows a analyst to define different components that may be in a hierarchical structure, as such they indirectly let an analyst define a hierical structure, but they do not use this information opting for flat tables and list at later stages of the analysis. A overview of the support for this suggestion may be seen in table 7.5.

7.6.1 Communication

All current solutions except STPA prototype supports defining communications between the different levels in a control structure, but this information is not used that much, as the later stages of the analysis is flat. All of these solution allows the analyst to define control action that is used to different levels. XSTAMPP and STPA Prototype allows the analyst to define process model variables, that tells what the current component thinks of the state of the system.

7.6.2 Abstraction views

None of the solutions presented in this chapter directly supports abstraction of different levels. But feedback, control actions and process model variables may be used by the analyst to define abstractions.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The objective of this thesis was to look at a new safety engineering method, understand how this method could be improved with the help of software solutions, and if possible implement a software solution for achieving this. The following are the contributions and conclusions from this thesis.

8.1.1 Conclusion

The possibility for stpa to be used in agile software development environment was explored throughout this thesis. By identifying what functionality a software tool could provide in such an environment, there was created a full list of requirements. These requirements was used to implement a system, that was then compared to other software solutions. While the system lacked in basic functionality, as it supported the extended STPA methods there were some favourable results.

8.2 Future Work

The future of this work is a full implementation of all the requirements presented in this thesis. While the implementation allows for implementation of a STPA step 1 analysis, it is not mature, and is not suggested to be used by professional teams until it is more mature. The following are some of the functionality that can be further developed to properly separate the

implementation from others.

Agile software team support: As discussed in this thesis, STPA is a linked analysis, as such a software solution could make use of this to provide full version control with information to the analyst for what would need to be re-analysed based on previous changes. There should also be performed proper testing of the solution in team environments, as the solution is web based and would allow multiple users in a team to work on a project at the same time, or using the power of version control to have different versions of a project where they may merge in changes.

Automation: While the current implementation takes use of some of the suggestions presented by Thomas for automating STPA analysis, more could be done. The current logical simplification is a manual method, that could be changed for automatic and abstractions are not implemented.

Traceability: The current implementation implements traceability, but more may be done to properly inform the analyst of how the system is connected.

As this implementation focused on functionality, there should also be work done to properly achieve the quality requirements set in this thesis.

Bibliography

- [1] Asim Abdulkhaleq and Stefan Wagner. "Open Tool Support for System-Theoretic Process Analysis". In: (2014).
- [2] Agile Alliance. *Manifesto for Agile Software Development*. URL: <http://agilemanifesto.org> (visited on 11/01/2016).
- [3] Safety-Critical Systems Research Lab Team of ZHAW Zurich University of Applied Sciences. *Partnership for Systems Approaches to Safety and Security (PSASS) - STAMP Tools*. URL: <https://psas.scripts.mit.edu/home/2016-2/> (visited on 06/20/2018).
- [4] Safety-Critical Systems Research Lab Team of ZHAW Zurich University of Applied Sciences. *SAHRA - STPA based Hazard and Risk Analysis*. URL: www.sahra.ch (visited on 06/20/2018).
- [5] Paul Clements, Rick Kazman, and Len Bass. *Software Architecture in Practice*. 2013.
- [6] Mike Cohn. *Succeeding with agile: software development using Scrum*. Pearson Education, 2010.
- [7] N Leveson and J Thomas. "An STPA Primer Version 1". Unfinished. 2013.
- [8] Nancy Leveson. "A new accident model for engineering safer systems". In: *Safety science* 42.4 (2004), pp. 237–270.
- [9] Nancy Leveson. "A systems model of accidents". In: *Proceedings of the 20th International System Safety Conference*. International Systems Safety Society Unionville, USA. 2002, pp. 476–486.
- [10] Nancy Leveson. *Engineering a Safer and More Secure World*. Presentation. 2016.
- [11] Nancy Leveson. *Engineering a safer world: Systems thinking applied to safety*. 2011.
- [12] Niklas Ludvigsen. "Preliminary study of agile safety and security analysis using STPA and STPA-Sec". TDT4501 Project. 2016.

- [13] NTNU. *TDT4501 - Datateknologi, fordypningsprosjekt*. URL: <https://www.ntnu.edu/studies/courses/TDT4501#tab=omEmnet> (visited on 06/21/2018).
- [14] Briony J Oates. *Researching information systems and computing*. Sage, 2005.
- [15] Information technology Promotion Agency. *STAMP Workbench*. URL: https://www.ipa.go.jp/sec/stamp_wb/manual/index.html# (visited on 06/21/2018).
- [16] Armin Ronacher. *Flask Foreword*. URL: <http://flask.pocoo.org/docs/1.0/foreword/> (visited on 06/10/2018).
- [17] Armin Ronacher. *Flask Welcome*. URL: <http://flask.pocoo.org/> (visited on 06/11/2018).
- [18] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.
- [19] RM Studio. *RM Studio - Risk Management*. URL: <https://www.riskmanagementstudio.com> (visited on 06/20/2018).
- [20] RM Studio. *RM Studio - STPA*. URL: <https://www.riskmanagementstudio.com/features/stpa> (visited on 06/20/2018).
- [21] Dajiang Suo. "Tool-assisted hazard analysis and requirement generation based on STPA". PhD thesis. Massachusetts Institute of Technology, 2016.
- [22] WTFForms Team. *WTFForms Crash Course*. URL: https://wtforms.readthedocs.io/en/stable/crash_course.html (visited on 06/11/2018).
- [23] XSTAMPP Team. *XSTAMPP For Safety Engineering of Software Intensive Systems*. URL: <http://www.xstamp.de/> (visited on 10/31/2016).
- [24] XSTAMPP Team. *XSTAMPP For Safety Engineering of Software Intensive Systems Installation Guide*. URL: <http://www.xstamp.de/Download.html> (visited on 10/31/2016).
- [25] Dr. John Thomas. *Intro to Systems Theoretic Process Analysis (STPA)*. Presentation. 2016.
- [26] John Thomas. "Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis". PhD thesis. Massachusetts Institute of Technology, 2013.
- [27] John Thomas and Dajiang Suo. *A Tool-Based STPA Process*. Presentation. 2013.

- [28] Yang Wang and Stefan Wagner. "Toward Integrating a System Theoretic Safety Analysis in an Agile Development Process." In: *Software Engineering (Workshops)*. 2016, pp. 156–159.
- [29] William Young and Nancy Leveson. "Systems thinking for safety and security". In: *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM. 2013, pp. 1–8.
- [30] William Young and Nancy G Leveson. "An integrated approach to safety and security based on systems theory". In: *Communications of the ACM* 57.2 (2014), pp. 31–35.
- [31] William E Young Jr. "STPA-SEC for cyber security mission assurance". In: *Eng Syst. Div. Syst. Eng. Res. Lab* (2014).

Appendix A

Setup

This document describes how to set up the system.

The prerequisites to run the system are:

- Python 2.7
- (optional) Virtualenv

A.1 Start the system

1. (optional) Create and activate a Virtualenv:

```
https://virtualenv.pypa.io/en/stable/userguide/
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Run the system:

```
python run.py
```